Scilab Textbook Companion for Coulson And Richardson's Chemical Engineering, Volume 2 by J. M. Coulson, J. F. Richardson, J. R. Backhurst And J. H. Harker[1]

Created by
Ankit Mishra
B.Tech
Chemical Engineering
IIT-BHU, VARANASI
College Teacher
R.S.singh
Cross-Checked by

May 20, 2016

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Coulson And Richardson's Chemical Engineering, Volume 2

Author: J. M. Coulson, J. F. Richardson, J. R. Backhurst And J. H. Harker

Publisher: Elsevier India

Edition: 5

Year: 2006

ISBN: 9788181471444

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

Chapter 1

Particulate Solids

Scilab code Exa 1.1 Surface mean diameter

```
clear;
clc;
printf("\n Example 1.1");
//Given size analysis of a powdered material
d=[1,101];//diameter of the powdered particles
x=[0,1];//mass fractions of the particles
plot2d(d,x,style=2,rect=[0,0,120,1])
xtitle("size analysis of powder","particle size(um)"
    ,"mass fraction(x)")
d=100*x+1; // from the given plot
//calculation of surface mean diameter
function[ds]=surface_mean_diameter(x0,x1)
    ds=1/(integrate('1/(100*x+1)','x',x0,x1))
    funcprot(0)
endfunction
ds=surface_mean_diameter(0,1);//deduced surface mean
    diameter according to def.
printf("\n The surface mean diameter is %fum",ds);
```

Scilab code Exa 1.2 Surface and mass distribution curve and surface mean diameter

```
clear;
clc;
printf("\n Example 1.2");
//from given differential eq we get these functions
//particle number distribution for the size range
   0-10um


//n=0.5*d^2;
//const of integration is0 since at n=0,d=0

//particle number distribution for the size range
   10-100um
//n=83-(0.33*(10^(5))*d^(-3))
//c2=83,since at d=10um,n=50

//number distribution plot for the powdered material
    of size range 0-100um
function[n]= number_distribution(d)
    if(d<=10) then
        n=0.5*d^2;
    else
        n=83-(0.33*(10^(5))*d^(-3));
        end
        funcprot(0)
endfunction
d=0;
while(d<=100)
  n=number_distribution(d);
    plot(d,n,"+-");
    d=d+1;
end
xtitle("number_distribution_plot","diameter(um)","
   number distribution");
ps=[0 6.2 9.0 10.0 11.4 12.1 13.6 14.7 16.0 17.5
```

```
19.7 22.7 25.5 31.5 100];
function[n1]=difference(i)
//ps=[0 6.2 9.0 10.0 11.4 12.1 13.6 14.7 16.0 17.5 19.7 22.7 25.5 31.5 10];
//according to the given particle sizes particle sizes are in um
    n1=number_distribution(ps(i+1))-number_distribution(ps(i));
    funcprot(0);
endfunction
function[da]=average(i)
    da= (ps(i+1)+ps(i))/2;
    funcprot(0);
endfunction
tot_n1d12=0;
tot_n1d13=0;
i=1;
for i=1:14
        tot_n1d12=tot_n1d12+difference(i)*(average(i))^2;
        tot_n1d13=tot_n1d13+difference(i)*(average(i))^3;
end
printf("\n tot_n1d12 =%d \n tot_n1d13=%d",tot_n1d12,tot_n1d13);
function[s]=surface_area(j)
    s=(difference(j)*(average(j))^2)/tot_n1d12;
    funcprot(0);
endfunction
su=0;
j=0;
xset('window',1);

plot(0,0,"o-");
for j=1:14
    su=su+surface_area(j);
    plot(ps(j+1),su,"o-");
end
```

```
xtitle("surface area and mass distribution plot","diameter(um)","surface area or mass distribution");
//mass distribution plot
function[x]=mass_distribution(k)
    x=(difference(k)*(average(k))^3)/tot_n1d13;
    funcprot(0);
endfunction
ma=0;
k=0;
plot(0,0,"+-");
for k=1:14
    ma=ma+mass_distribution(k);
    plot(ps(k+1),ma,"+-");
end
//evaluating surface mean diameter
function[d]=surface_mean_diameter(l)
    e=0;
    for l=1:14
        n=(mass_distribution(l)/average(l));
        e=e+n;
    end
d=1/e;
    funcprot(0);
endfunction
printf("\nthe surface mean diameter is: %fum",surface_mean_diameter());
```

Scilab code Exa 1.3 variation of mixing index with time

```
clear all;
clc;
printf("\n Example 1.3");
p=0.20;//components analysed represents 20 percent of the mixture by mass
```

```
//for a completely unmixed system
so=p*(1-p);
//for a completely random mixture :
n=100;//Each of the sample removed contains 100 particles
sr=p*(1-p)/n;
s=[0.025 0.006 0.015 0.018 0.019];
time_secs=[30 60 90 120 150];
printf("\n degree of mixing is :\n")
function[b]=degree_of_mixing()
for i=1:5
    b(i)=(so-s(i))/(so-sr);
    disp(b(i));//b is the degree of mixing
end
    return b;
funcprot(0)
endfunction
plot2d(time_secs,degree_of_mixing(),style=3)
xtitle("degree of mixing curve","time_secs","degree_of_mixing")
//plot of sample variance vs time(secs)
xset('window',1)
plot2d(time_secs,s,style=2)
xtitle("sample variance curve","time_secs","sample variance")
//from the graph the maxima is at 60 secs
```

Scilab code Exa 1.4 minimum apparent density for seperation

```
//minimum size of the particle in the mixture of quartz and galena(mm)
clear all;
clc;
printf("\n Example 1.4");

```

```
//maximum size of the particle(mm)
d_max=0.065;
//minimum size of the particle(mm)
d_min=0.015;
//density of quartz(kg/m^3)
p_quartz=2650;
//density of galena (kg/m^3)
p_galena=7500;
//minimum density of the particle which will give this seperation
//When stoke's law is applied the required density is as given below
function[d]=stoke_required_density()
    p=poly([0],'p');
    d=roots((p-7500)-(p-2650)*(d_max/d_min)^2);
    funcprot(0);
endfunction
d=stoke_required_density();
printf("\n required density is = %d kg/m^3",d);
//When Newton's law is applied then the required density is as given below
function[e]=newton_required_density()
    r=poly([0],'r');
    e=roots((r-7500)-(r-2650)*(d_max/d_min));
    funcprot(0);
endfunction
e=newton_required_density();
printf("\nrequired density is by newton law =%d kg/m^3",e);
```

Scilab code Exa 1.5 efficiency of collection for dust

```
clear;
clc;
printf("\n Example 1.5");
```

```
//efficiency of the collector for different size ranges
efficiency_1=45;//in percentage for the size range of 0-5um
efficiency_2=80;//in percentage for the size range of 5-10um
efficiency_3=96;//in percentage for the size range greater than 10um

//mass percent of the ndust for various size range
mass_1=50; //in percentage for the size range of 0-5 um
mass_2=30; //in percetage for the size range of 5-10 um
mass_3=20; //in percentage for the size range greater than 10um
// on the basis of 100kg dust
mass_retained_1=0.45*50;//mass retained(kg) in the size range of 0-5um
mass_retained_2=0.80*30;//mass retained(kg) in the size range of 5-10um
mass_retained_3=0.96*20;//mass retained(kg) in the size range greater than10um
overall_efficiency=0.45*50+0.80*30+0.96*20;
printf("\n the overall efficiency is =%f", overall_efficiency);
```

Scilab code Exa 1.6 Overall efficiency of collector

```
clear;
clc;
printf("\n Example 1.6");
//To calculate mass flow of the dust emitted
mass_1=10;//in percentage in the size range of 0-5um
mass_2=15;//in percentage in the size range of 5-10
```

```
um
mass_3=35;//in percentage in the size range of 10-20um
mass_4=20;//in percentage in the size range of 20-40um
mass_5=10;//in percentage in the size range of 40-80um
mass_6=10;//in percentage in the size range of 80-160um
efficiecny_1=20;//in percentage in the size range of 0-5um
efficiency_2=40;//in percentage in the size range of 5-10um
efficiency_3=80;//in percentage in the size range of 10-20um
efficiency_4=90;//in percentage in the size range of 20-40um
efficiency_5=95;//in percentage in the size range of 40-80um
efficiency_6=100;//in percentage in the size range of 80-160um
dust_burden=18;//in g/m^3 at the entrance
//taking 1m^3 as the basis of calculation
total_mass_retained=18*(0.1*0.20+0.15*0.40+0.35*0.80+0.2*0.9+0.1*0.95+0.1*1);
printf("\ntotal mass retained =%fg",total_mass_retained);
total_efficiency=(total_mass_retained/18)*100;
printf("\ntotal efficiency is =%f",total_efficiency);
total_mass_emitted=18-total_mass_retained;
printf("total mass emitted is:%fg",total_mass_emitted);
t=18*(0.1*0.80+0.15*0.60+0.35*0.20);
printf("\ntotal mass emitted less than 20um is %fg",t);
e=t*100/total_mass_emitted;
```

```
printf("\nThe efficiency of particles emitted is %f"
    ,e);
//gas flow is 0.3m^3/sec
f=0.3*total_mass_emitted;
printf("\nmass flow rate is:%fkg/sec",f);
```

Scilab code Exa 1.7 Estimation of particle size

```
clear;
clc;
printf("\n Example 1.7");
Ai=(%pi/4)*(0.075)^2;//cross sectional area at the
    gas inlet in m^2
do=0.075;//gas outlet diameter in m
p=1.3;//gas density in kg/m^3
Z=1.2;//height of the seperator in m
dt=0.3;//seperator diameter in m
v=1.5;//gas entry velocity in m/sec
G=(Ai*v*p);//mass flow rate of the gas in kg/sec
printf("\n cross sectional area at the gas inlet is
    %fm^2",Ai);
printf("\ngas outlet diameter is %fm",do);
printf("\ngas density is %fkg/m^3",p);
printf("\nheight of the seperator is %fm",Z);
printf("\nseperator diameter is%fm",dt);
printf("\nmass flow rate of the gas is %fkg/sec",G);
function[u]=terminal_vel()
    u=0.2*(Ai)^2*(do)*p*9.8/(%pi*Z*(dt)*G);//
        velocity is in m/sec
    funcprot(0);
endfunction
u=terminal_vel();
printf("\nthe terminal velocity of the smallest
    particle retained by the seperator =%fm/sec",u);
function[d]=particle_diameter(u)
```

```
    u=terminal_vel();
    n=0.018*10^(-3);//viscosity in mNs/m^2
    ps=2700;//density of the particle in kg/m^3
    d=((u*18*n)/(9.8*(ps-p)))^(0.5);//particle size in um
    funcprot(0);
endfunction
u=terminal_vel();
d=particle_diameter(u);
do=d*10^6;
printf("\n particle diameter by the stoke law is %fum",do);
```

Chapter 2

Particle size reduction and enlargement

Scilab code Exa 2.1 Consumption of energy

```
clear;
clc;
printf("\n Example 2.1");
//Computing energy required in particle size
   reduction by Rittinger's law
//energy required in crusing is given by E =Kr.fc
   ((1/L2)-(1/L1))
//Given : Energy required in crushing particles from
    50mm to 10mm is        13.0kw/(kg/sec).

Cr=13.0*(50/4);//Cr = Kr*fc

//Energy required to crush the particles from 75mm
   to 25mm
E = Cr*((1/25)-(1/75));
printf("\n The energy required in crushing the
   materials from average particle size of 75mm to
   25mm by Rettingers law is %fkj/kg",E)

```

```
//Computing the energy required for crushing by Kick
    's law
//E = (Kk*fc)*ln(L1/L2) by Kick's law

Ck = (13.0)/log(50/10);//Ck = Kk*fc

//Energy required to crush the material from 75mm to
     25mm
Ek = Ck*log(75/25);
printf("\nThe energy required for crushing the
    material from average particle size of by Kicks
    law 75mm to 25mm is %fkj/kg \n",Ek);
printf("\n The size range required is that for
    coarse crushing and Kicks law more closely
    relates the energy required for plastic
    deformation before fracture occurs so the energy
    calculated as that by Kicks law will be taken as
    the more reliable result\n");
```

Scilab code Exa 2.2 Maximum size of the particle

```
clear;
clc;
printf("\n Example 2.2");
//Calculating the maximum size of the particle that
    can be fed to the rollers.
//Given angle of nip = 31degree
//Given diameter of the crushimg rolls = 1m
//Distance between the crushing rolls is 12.5mm
r1 = 0.5;//size of crushing rolls is in meters
b = 0.00625;//Distance between the crushing rolls is
     0.0125mm
r2 = (r1 + b)/(cos((%pi/180)*15.5))-0.5;
printf("\nThe maximum size of the particles which
    should be fed to the rollers : %d mm",r2*10^3);
```

```

//Calculating the throughput at 2.0 Hz when the
    actual capacity of the machine is 12%.
//Working face of the rolls are 0.4m long
//bulk density of the feed is 2500kg/m^3
printf("\nThe cross sectional area for flow is %.3f
    m^2",0.0125*0.4);
printf("\nThe volumetric flow rate is %.2f m^3/sec"
    ,2.0*0.005);
printf("\nThe actual throughput is %d kg/sec"
    ,0.010*12*2500/100);
```

Scilab code Exa 2.3 Proposed modifications in ball mill

```
clear;
clc;
printf("\n Example 2.3");
//Given diameter of the ball mill is 1.2meters
//Speed of rotation is 0.80Hz

//for small particles effective radius is 0.6meters
//critical speed of the rotation
g=9.80;       //acceleration due to gravity is in m
    ^2/sec.
r=0.6;        //effective radius of rotation is in
    meters.
w = sqrt(g/r);
printf("\nThe critical speed of the rotation is %.2f
     rad/sec",w);
f=w/(2*%pi); //f is the frequency of the rotation
    and is in Hz
printf("\nThe critical frequency of the rotation is
    %fHz\n",f);
optimum_frequency = 0.6*f;
printf("The optimum frequency of the rotation is %.2
```

```
    f Hz",optimum_frequency);
printf("\nGiven frequecy of the rotation is 0.80Hz\n
    ");
printf("The frequency of the rotation should be
    halved\n");
printf("Therefore the optimal frequency is half the
    critical frequency");
```

Chapter 3

Motion of particles in a fluid

Scilab code Exa 3.1 terminal velocity

```
clear all;
clc;
printf("\n Example 3.1");
    d = 0.00040;    //Diameter of the particle in m
    p1 = 820;       //Density of the fluid in kg/m^3
    meu = 0.01;     //Viscosity of the fluid in N s/m
       ^2
    p2 = 7870;     //Density of steel in kg/m^3
    g = 9.81;      //Acceleartion due to gravity in m
       /sec^2
//Computation of terminal velocity of a spherical
   particle
function[x]=galileo_number()

    x = (2*d^3*(p2-p1)*p1*g)/(3*(meu)^2); //x = (Ro/
       pu^2)*Re^2
    funcprot(0);
endfunction
x = galileo_number();
printf("\n The value of (Ro/pu^2)/Re^2 is  %f\n",x);
```

```
//From table 3.4 log(x) corresponds to log(Re)=0.222
Re = 1.667;
function[u]=teminal_velocity()
    u=(Re*meu)/(p1*d);
    funcprot(0);
endfunction
 u = teminal_velocity();
 printf("\n The terminal velocity of the steel ball
    is %.3f m/sec or %.0f mm/sec",u,u*1000);
```

Scilab code Exa 3.2 Estimation of galena

```
clear;
clc;
printf("\n Example 3.2");
u_water = 5*10^(-3);  //The flow velocity of the
   water in m/sec
p_galena = 7500;      //The density of galena is in
   kg/m^3
p_limestone = 2700;   //The density of limestone is
   in kg/m^3
viscosity = 0.001;    //The viscosity of water in N
   s/m^2

//calculating maximum value of reynold's number
   considering 5mm particle size
Re_max = (u_water*1000*0.0001)/(viscosity);
printf("\n The maximum permissible value of Re is %f
   ",Re_max);

//maximum particle size of galena which will be
   carried away by water
d = sqrt((u_water*(18*viscosity))/((7500-1000)*9.81)
   );
```

```
printf("\nmaximum particle size of galena which will be carried away by water is %.1f um",d*10^(6));

//maximum particle size of limestone which will be carried away by water
d1 = sqrt((u_water*(18*viscosity))/((2700-1000)*9.81));
printf("\nmaximum particle size of limestone which will be carried away by water is %.1f um",d1*10^(6));


//From the given data 43% galena and 74% limestone will be removed .
//Given that in the feed there is 20% galena and 80% limestone
//Assuming 100g feed
printf("\n\nIn the overflow:");
printf("\nAmount of galena is %fg",(20*0.43));
printf("\nAmount of limestone is %fg",(80*0.74));
printf("\nconcentration of galena is %.1f per cent",(20*0.43*100)/(20*0.43+80*0.74));
printf("\n\concentration of galena is %fper cent",(80*0.74*100)/(20*0.43+80*0.74));
printf("\n\nIn the underflow:")
printf("\nconcentration of galena is %.1f percent",(20*(1-0.43)*100)/(20*(1-0.43)+80*0.26))
printf("\nconcentration of limestone is %.1f per cent",(80*0.26*100)/(20*0.57+80*0.26))
```

Scilab code Exa 3.3 terminal velocity

```
clear;
clc;
printf("\n Example 3.3");
```

```
min_area =6*10^(-6);     //minimum area of mica
    plates in m^2
max_area =6*10^(-4);     //maximum area of mica
    plates in m^2
p_oil = 820;             //density of the oil in kg/m
    ^3
Viscosity = 0.01;       //Viscosity is in N s/m^2
p_mica = 3000;          //Density of mica in kg/m^3

printf("\n      smallest particles
                              largest particles");
printf("\nA:     %fm^2
                                                        %fm^2
     ",min_area,max_area);
printf("\ndp:   %fm
                                                          %fm
    ",sqrt(4*min_area/(%pi)),sqrt(4*max_area/(%pi)));
printf("\ndp^3:%f*10^(-8)m^3
                                              %f*10^(-5)m^3
    ",sqrt(4*min_area/(%pi))^(3)*10^(8),sqrt(4*
    max_area/(%pi))^(3)*10^5);
printf("\nv:     %f*10^9m^3
                                                %f*10^7m^3
    ",0.285*sqrt(4*min_area/(%pi))^(3)*10^(9),sqrt(4*
    max_area/(%pi))^(3)*(10^7)*0.0285);
printf("\nk:     %f
                                                          %f
    ",0.285,0.0285);

x1 = (4*0.285/(%pi*0.01^2))*(3000-820)
    *(820*2.103*10^(-8)*9.81);
x2 = (4*0.0285/(%pi*0.01^2))*(3000-820)
    *(820*2.103*10^(-5)*9.81);
printf("\n(Ro/pu^2)Reo^2 is %d for the smallest
    particles and %d for the largest particles",x1,x2
    );
//From table 3.4 Re for smallest particle is 34.9
    and that for the largest particle is 361
```

```
u1 = 34.9*0.01/(820*2.76*10^(-3));
printf("\nTerminal velocity for the smallest
    particle is %.3f m/sec",u1);
u2 = 361*0.01/(820*2.76*10^(-2));
printf("\nTerminal velocity for the largest particle
     is %.3f m/sec",u2);
printf("\n\n Thus it is seen that all the particles
    settle at approximately the same velocity");
```

Scilab code Exa 3.4 Approximate distance travelled

```
clear;
clc;
printf("\n Example 3.4");
v_particle = 6;  //velocity of the particle in m/sec
v_water = 1.2;   //veloicity of the water in m/sec
v_rel = v_particle - v_water;//relative velocity of
    particles relative to the fluid in m/sec
Re1 = 6*10^(-3)*v_rel*1000/(1*10^(-3));//Re1 is the
    reynold's no.
printf("\nReynold no. is %d",Re1);

//When the particle has been retarded to a velocity
    such that Re=500
ydot = (v_rel*500)/Re1;
printf("\nParticle velocity is %.3f m/sec\n",ydot);
c = 0.33/(6*10^(-3))*(1000/2500);
f = sqrt((3*6*10^(-3)*(2500-1000)*9.81)/1000);
function[y]=Fa(t)
    y = (-1/22)*(log(cos(0.517*22*t) + 4.8/0.517*sin
        (0.517*22*t)));
    funcprot(0);
endfunction

function[yd]=deriv(t)
```

```
    yd= -0.083+(0.517*(9.28*cos(11.37*t) - sin(11.37*t))/(cos(11.37*t) + 9.28*sin(11.37*t)));
    funcprot(0);
endfunction

function[ydd]=double_deriv(t)
    ydd = -0.517*(11.37)^2*(9.28*cos(11.37*t) - sin(11.37*t))/(cos(11.37*t) + 9.28*sin(11.37*t));
    funcprot(0);
endfunction


told = 0;
while 1
    tnew = told - deriv(told)/double_deriv(told);
    if (tnew == told) then
        y = Fa(told);
        d = y;
        printf("\nThe distance moved with speed less than 0.083m/sec is %.3fm",d);
        t=told;
        printf("\n The time taken by particle to move this distance is %.3fsec",t);
        break;
    end
    told = tnew;
end


printf("\nThe distance moved by the particle relative to the walls of the plant %.3fm",1.2*t - d);
```

Scilab code Exa 3.5 maximum size of crystals

```
clear;
clc;
printf("\n Example 3.5");
//rate of dissolution of salt
function[x]=dissolution(d)
   x = (3*10^(-6))-(2*10^(-4)*3.406*10^(5)*d^2);
   funcprot(0)
endfunction

//rate of falling of the particle in stokes law region
function[y]=rate_h(d)
    y = 3.406*d^(2)/(-3*10^(-6)-68.1*d^2);//y is in m/sec
    funcprot(0);
endfunction

printf("\n By trial and error the solution for d is 0.88 mm");
printf("\n The rate of dissolution is %f ",dissolution(8.8*10*(-4)));
printf("\n The rate of falling of the particle is %f m/sec",rate_h(8.8*10^(-4)));
```

Chapter 4

Flow of fluids through granular beds and packed columns

Scilab code Exa 4.1 pressure calculation

```
clear;
clc;
printf("\n Example 4.1");
//Calculating modified reynold's no.
a = 800;                    //it is in m^2/m^3
Product_rate = 0.5;         //it is in g/sec
Reflux_ratio = 8;
Vapour_rate  = 4.5;         //it is in g/sec
G = (4.5*10^(-3))/((%pi/4)*(0.1^2));// units are in kg/m^2.sec
meu = 0.02*10^(-3);                 //units are in Ns/m^2
e = 0.72;

Re1 = G/(800*0.28*0.02*10^(-3));
printf("\n The modified reynolds no. is %d",Re1);

x = 4.17/Re1 + 0.29; //x = R1/(pu1^2)
printf("\n The value of R1/(pu1^2) is %f",x);
```

```
l = 16*0.15;            //in meters
//Solving the integral integral of(pdP)from Pc toPs
   =(R1/pu^2)*S*(1-e)*G^2*l/e^3

Pc = poly([0],'Pc');
y = roots(151.3-(4.73*10^(-5)*(8000^2-Pc^2)));
printf("\n The value of Pc is %dN/m^2",y(1));
```

Chapter 5

Sedimentation

Scilab code Exa 5.1 Minimum area of thickener

```
clear;
clc;
printf("\n Example 5.1");
//Basis 1 kg of solids
feed_rate_solid = 1.33;  //Mass rate of feed of solids in kg/sec
U = 1.5;                 //Mass rate of solids in the underflow in kg/sec
Y = [5.0 ; 4.2; 3.7 ; 3.1; 2.5];
printf("\n concentration(Y) (kg water/kg solids):\n");
printf("%.1f\n",Y);
printf("\n water to overflow (Y-U) (kg water/kg solids):\n");
O = Y - 1.5;//Amount of water to overflow in kg water/kg solids
printf("\n %.1f\n",O);
Uc = [2.00*10^(-4);1.20*10^(-4);0.94*10^(-4);0.70*10^(-4);0.50*10^(-4)];
printf("\n sedimentation rate uc (m/sec):\n");
printf("%f  \n",Uc);
```

```
X =[1.75*10^4;2.25*10^4;2.34*10^4;2.29*10^4;2.00*10^4]; //X = (Y-U)/Uc
printf(" \n\n(Y-U)/Uc (s/m):\n");
printf("\n %d\n",X);
z = max(X); //prints the maximum value of X
printf("\nMaximum value of (Y-U)/Uc = %ds/m",z);

//Calculating the require darea of the thickener
A = z*1.33/1000; //1.33 is the mass feed rate of solids in kg/sec
//1000 is the density of water in kg/m^3
printf("\n The required area of the thickener is :%.2fm^2\n",A);
```

Scilab code Exa 5.2 Sedimentation velocity and solids flux

```
clear;
clc;
printf("\n Example 5.2");
//Area of the tank required to give an underflow concentration of 1200kg/m^3 for a feed rate of 2 m^3/min

//Initial height of slurry in the tank
H = [900;800;700;600;500;400;300;260;250;220;200;180];

uc = [13.4;10.76;8.6;6.6;4.9;3.2;1.8;1.21;1.11;0.80;0.60;0.40];

i=1;
while i<13
    c(i)=200*900/H(i);
    x(i)=1000*(1/c(i)-1/1200);
```

```
    sed(i) = c(i)*uc(i)/(1000*60);
    y(i)= uc(i)*10^(-3)/((1/c(i)-1/1200)*60);
    z(i) = 1/y(i);
    i=i+1;
end
printf("\nH(mm)");
printf("\n%d",H);
printf("\n c(kg/m^3):\n");
printf("%d\n",c);
printf("Sedimentation flux(kg.s/m^2):\n");
printf("%.4f\n",x);
printf("uc/(1/c-1/1200)\nkg.sec/m^2:\n");
printf("%.4f\n",y);
printf("1000*(1/c-1/cu)\nm^3/kg*10^3\n");
printf("%.3f \n",x);
printf("\n\n(1/c-1/1200)/uc\n m^2.kg/sec\n");
printf("%.1f\n",z);
m1=max
   ([18.7;20.1;21.3;22.7;23.8;26.0;27.8;30.3;30.0;29.2;27.8;25.0])
   ;
printf("\n\nthe maximum value of (1/c-1/1200)/uc is
   %.1f m^2*kg/s",m1);
A = 2*200*30.3/60;
printf("\n The area required is A = Qc[(1/c-1/cu)/uc
   ]max = %dm^2",A)
```

Scilab code Exa 5.3 Rate of deposition and maximum flux

```
clear;
clc;
printf("\t Example 5.3 ");
//Assumption: Resistance force F on an isolated
   sphere is given by Stoke's law:F = 3*%pi(meu)d*u

C = poly([0],'C');
```

```
x=roots(-4.8*C+(1-C));
printf("\n concentration is:%.3f",x);

//terminal falling velocity u can be calculated by force balance
//u = d^2*g/(18*meu)*(ps-p)
function[u]=terminal_velocity()
    d = 10^(-4);    //diameter is in meters
    g = 9.81;       //acceleration due to gravity is in m/sec^2
    meu = 10^(-3); //viscosity is in N.s/m^2
    ps = 2600;      //density is in kg/m^3
    p = 1000;       //density is in kg/m^3

    u = (d^2)*g*(ps-p)/(18*meu);
    funcprot(0);
endfunction


function[si]=si_max()
    u=terminal_velocity()
    printf("\n The terminal falling velocity is %.5f m/sec",u);
    si=u*x*(1-x)^(4.8);
    funcprot(0);
endfunction
si = si_max();
printf("\nThe maximum value is %f*10^(-4) m^3/m^2sec",si*10^4)
```

Chapter 6

Fluidisation

Scilab code Exa 6.1 minimum fluidising velocity

```
clear;
clc;
printf("\n Example 6.1");
//Calculating minimum fluidisation velocity

//Calculating Galileo number
function[Ga]=Galileo_number()
    d = 3*10^(-3);  //particle size is in meters
    p = 1100;       //density of liquid is in kg/m^3
    ps = 4200;      //density of spherical particles is in kg/m^3
    g = 9.81;       //acceleration due to gravity is in m/sec^2
    u = 3*10^(-3);  //viscosity is in Ns/m^2
    Ga = d^3*p*(ps-p)*g/u^2;
    funcprot(0);
endfunction
Ga = Galileo_number();
printf("\nGalileo number = %f*10^5",Ga*10^(-5));

//Calculating Re mf
```

```
Remf = 25.7*(sqrt(1+5.53*10^(-5)*(1.003*10^5))-1);
printf("\nValue of Remf is %d",Remf);

umf = Remf*(3*10^(-3))/(3*10^(-3)*1100);
printf("\nminimum fluidisation velocity is %.1f mm/
    sec",umf*1000);
```

Scilab code Exa 6.2 fluidisation and transport of particles

```
clear;
clc;
printf("\n Example 6.2");
//Calculating voidage by considering eight closely
    packed spheres of diameter d in a cube of size 2d
printf("\n (a)");
function[e]=voidage()
    d = 1*10^(-4);     //diameter is in meters
    meu = 3*10^(-3);   //viscosity is in Ns/m^2
    ps = 2600;         //density is in kg/m^3
    p = 900;          //density is in kg/m^3
    e = [8*d^(3)-8*(%pi/6)*d^(3)]/(8*d^(3));
    funcprot(0);
endfunction
e = voidage();
printf("\nvoidage = %.2f",e);

//Calculating minimum fluidisation mass flow rate

function[Gmf]=min_fluidis_vel()
    e = voidage();
    d = 1*10^(-4);     //diameter is in meters
    meu = 3*10^(-3);   //viscosity is in Ns/m^2
    ps = 2600;         //density is in kg/m^3
    p = 900;          //density is in kg/m^3
    g = 9.81;          //acceleration due to gravity
```

```
is in m/sec^2
    Gmf = 0.0055*(e)^(3)/(1-e)*(d^2)*p*(ps-p)*g/meu;
    funcprot(0);
endfunction
Gmf = min_fluidis_vel();
printf("\nminimum fluidisation velocity is %.3f kg/m^2sec",Gmf);


printf("\n (b)");
function[u]=terminal_velocity()
    e = voidage();
    d = 1*10^(-4);    //diameter is in meters
    meu = 3*10^(-3);  //viscosity is in Ns/m^2
    ps = 2600;        //density is in kg/m^3
    p = 900;         //density is in kg/m^3
    g = 9.81;         //acceleration due to gravity is in m/sec^2
    u = d^(2)*g*(ps-p)/(18*meu);
    funcprot(0);
endfunction
printf("\nterminal velocity is %.4fm/sec",terminal_velocity());

//Reynolds no for this Terminal velocity is
Re = (10^(-4)*0.0031*900)/(3*10^(-3));
printf("\nReynlds no =%.3f",Re);
printf("\nThe required mass flow rate is %.2f kg/m^2sec",terminal_velocity()*900);
```

Scilab code Exa 6.3 Voidage of the bed

```
// to calculate voidage of the bed
clear;
clc;
```

```
printf("\n Example 6.3");
function[Ga]=Galileo_number()
    d = 4*10^(-3);  //particle size is in meters
    p = 1000;       //density of water is in kg/m^3
    ps = 2500;      //density of glass is in kg/m^3
    g = 9.81;       //acceleration due to gravity is in m/sec^2
    u = 1*10^(-3);  //viscosity is in Ns/m^2
    Ga = d^3*p*(ps-p)*g/u^2;
    funcprot(0);
endfunction
printf("\nGalileo number = %.2f*10^5",Galileo_number()*10^(-5));

function[Re]=Reynolds_no()
    Ga = Galileo_number();
    Re = (2.33*Ga^(0.018)-1.53*Ga^(-0.016))^(13.3);
    funcprot(0);
endfunction
printf("\n The Reynolds no is %d",Reynolds_no());
v = Reynolds_no()*(1*10^(-3))/(0.004*1000);
printf("\nvelocity = %.2f m/sec",v);

n = poly([0],'n');
z = roots((4.8-n)-0.043*(Galileo_number())^(0.57)*(n-2.4));
printf("\nvalue of n is %.2f",z);

//voidage at a velocity is 0.25m/sec
e=0.1;
while 1
    enew = e -((0.25/0.45)-e^(2.42))/(-2.42*e^1.42);
    if (enew == e) then
        printf("\nVoidage is %.3f",e);
        break;
    end
    e=enew;
end
```

Scilab code Exa 6.4 slope of adsorption isotherm

```
clear;
clc;
printf("\n Example 6.4");
t = 250:250:2000; //time is in secs
y =
    [0.00223;0.00601;0.00857;0.0106;0.0121;0.0129;0.0134;0.0137];

i =1;
yo = 0.01442;
while i<9
    z(i) = y(i)/yo;
    y(i)=1-z(i);
    x(i)=log(y(i));
    i=i+1;
end
xtitle("slope of adsorption isotherm","Time(sec)","
    log(1-(y/yo))")
plot(t,x,"o-")
printf("\nFrom the graph the value of slope is %f"
    ,-0.00167);
Gm = 0.679*10^(-6);   //units are in kmol/sec
W = 4.66;             //units are in gram
b = poly([0],'b');
s = roots(-0.00167*4.66*b+0.679*10^(-6));
printf("\n b = %.4f kmol/kg",s*10^3);
```

Scilab code Exa 6.5 Coefficient of heat transfer between gas and the particles

```
clear;
clc;
printf("\n Example 6.5");
gas_flow_rate =0.2;          //units are in kg/m^2
c = 0.88;                    //specific heat capacity of air is kj/kg K
viscosity = 0.015*10^(-3);//viscosity is in Ns/m^2
d = 0.25*10^(-3);            //particle size is in meters
k = 0.03;                    //thermal conductivity is in W/m K
e = 0.57;                    //e is voidage

T = [339.5;337.7;335.0;333.6;333.3;333.2]; //temperature is in kelvins
deltaT = T - 333.2;
h = [0;0.64;1.27;1.91;2.54;3.81];
xtitle("temperature rise as a function of bed height","height above bed support(mm)","deltaT(K)");
plot(h,deltaT,'o-');

//Area under the curve gives the value of  the heat teansfer integral =8.82mm K

q = 0.2*0.88*(339.5-332.2);
printf("\n heat transferred = %.2f kw/m^2 of bed cross sectional area",q);

//Assuming 1m^3 volume
Vp = (1-0.57);        //Volume of particles is in m^3
printf("\n Volume of particles is %.2f m^3",Vp)
v1 = (%pi/6)*d^3;     //Volume of 1 particle in m^3
printf("\n Volume of 1 particle is %.2f*10^(-12) m^3",v1*10^(12));
printf("\n number of particles is %.2f*10^(10)",Vp/v1*10^(-10));

x =poly([0],'x');
```

```
h = roots(1100 - x*(1.03*10^4)*(8.82*10^(-3)));
printf("\n heat transfer coefficient = %.1f W/m^2",h);

//Nu = 0.11Re^(1.28)
Re = (0.2*0.25*10^(-3))/(0.015*10^(-3));
h1 = 0.11*(Re)^(1.28)*k/d;
printf("\n h = %.1f W/m^2 K",h1);
```

Scilab code Exa 6.6 Volumetric fraction of the bed carrying out evaporation

```
clear;
clc;
printf("\n Example 6.6");
cp = 0.85;                                  //specific heat capacity of the air
h = [0 0.625 1.25 1.875 2.5 3.75];          //height in mm
T=[339.5 337.7 335.0 333.6 333.3 333.2];//temperature in K
deltaT = T - 333.2;                         //temperature difference in kelvins
plot(h,deltaT,"o-");
xtitle("deltaT as a function of bed height","Height above bed support z(mm)","Temperature difference deltaT (K)");
//From the plot area under the curve is 6.31 K mm
sp = (6/(0.25*10^(-3)))*(0.5);  //sp is surface area per unit volume in m^2/m^3
G = 0.2;                        //in kg/m^2sec
Cp = 850;                       //Cp is in J/kg K
h1 = poly([0],'h1');
s = roots(0.2*850*6.3-h1*1.2*10^(4)*6.31*10^(-3));
printf(" \n Coefficient for heat transfer between
```

```
    the gas and the particles= %.1fW/m^2 K",s);

printf("\n Let the evaporation rate be 0.1 kg/sec at
     a temp difference = 50 degK");
mdot = 0.1;          //evaporation rate is 0.1 kg/sec
Latent_heat = 2.6*10^(6);
printf("\n The heat flow = %.1f*10^5 W",mdot*
    Latent_heat*10^(-5));
A=(2.6*10^5)/(14.1*50);
printf("\n the effective area of the bed A = %d m^2"
    ,A);
printf("\n The surface area of the bed = %d m^2"
    ,0.1*1.2*10^4);
printf("\n hence the fraction of the bed = %.2f"
    ,369/1200);
```

Chapter 7

Liquid filtration

Scilab code Exa 7.1 volume of filtrate collected per cycle

```
clear;
clc;
printf("\n Example 7.1");
//In the leaf filter filtration is at const pressure
    from the start
//V^2 + 2ALV/v = 2(-deltaP)A^2t/(ruv)

//In the filter press,a volume V1 of filtrate is
   obtained under const rate conditions in time t1,
   and filtration is then carried out at constant
   pressure.
//V1^2 + 2ALV1/v = 2(-deltaP)A^2t1/(ruv)
//and (V^2      V1^2 ) + 2AL/   (V      V1) = 2(   P )A
   ^2/ r    (t      t1)

//for the leaf filter
t2 = 300;           //t2 is in secs
V2 = 2.5*10^(-4);    //V2 is in m^3
t3 = 600;            //t3 is in secs
V3 = 4*10^(-4);      //V3 is in m^3
A = 0.05;            //A is in m^2
```

```
deltaP = -7.13*10^(4); //it is in N/m^2
//putting these values in above eq

a = [2*7.13*10^(4)*0.05^(2)*300 -2*0.05*2.5*10^(-4)
    ;2*7.13*10^(4)*0.05^(2)*600 -2*0.05*4*10^(-4)];
b = [(2.5*10^(-4))^2;(4*10^(-4))^2];
x = inv(a)*b;
y = [1/x(1);x(2)];
printf("\n L/  =%f*10^(-3)   and  r     = %f*10^(11)"
    ,y(2)*10^3,y(1)*10^(-11));

//for the filter press
V1 = poly([0],'V1');
s = roots(V1^2 + (2.16*y(2)*V1)-(4*10^(5)*2.16^2)/y
    (1)*180);
printf("\n the value of V1 = %fm^3",s(2));

//For a constant pressure period (t - t1)=900secs
//Calculting the total volume of filtrate
V = poly([0],'V');
d = roots((V^2-3.33*10^(-4))+(1.512*10^(-2)*(V
    -1.825*10^(-2))-5.235*10^(-6)*900));
printf("\n The value of V = %.3f m^3",d(2));

f = (4*10^(5)*(2.16)^2)/(7.13*10^(11)*(6.15*10^(-2)
    + 2.16*3.5*10^(-3)));
printf("\n The final rate of filtration is %.2f
    *10^(-5) m^3/sec",f*10^(5));

// Assuming  viscosity of the filtrate is the same
    as that of the wash-water
rw_400 = (0.25)*f;
printf("\n Rate of washing at 400 kN/m2 = %.1f
    *10^(-6) m^3/sec",rw_400*10^(6));

rw_275 = rw_400*(275/400);
printf("\n Rate of washing at 275 kN/m^2 = %.1f
    *10^(-6) m^3/sec",rw_275*10^6);
```

```
printf("\n Thus the amount of wash-water passing in 600s = %.3f m^3",600*rw_275);
```

Scilab code Exa 7.2 Effect on optimium thickness of the cake

```
clear;
clc;
printf("\n Example 7.2");
//The slurry contains 100kg whiting/m^3 of water
printf("\n Volume of 100 kg whiting = %f m^3",100/3000);
printf("\n Volume of cake = %f m^3",0.0333/0.6);
printf("\n Volume of liquid in cake = %f m^3",0.05556*0.4);
printf("\n Volume of filtrate = %.3f m^3",(1-0.0222));
printf("\n volume of cake/volume of filtrate v = %f",0.0556/0.978);
A = 10^(-4);                    //area in sq meters
deltaP = -1.65*10^(5);          //P is in pascals
l = 0.01;                       //length is in meters
vol_flow_rate = 2*10^(-8);          //Volume flow rate is in m^3/sec
u = 10^(-3);                    //vicosity is in Ns/m^2

r = poly([0],'r');
r1 = roots((10^4)*(2*10^(-8)*r)-1.65*10^(5)/(10^(-5)));
printf("\n r = %.2f*10^(13)/m^2",r1*10^(-13));

function[Lopt]=optimum()
    Lopt = 1.161*10^(-3)*(900)^(0.5);   //t = 900 secs
    funcprot(0);
endfunction
```

```
printf("\n optimum frame thickness = %.1f mm",2*optimum()*1000);

//total cycle time = 1.015L^2 + 900
//rate of cake production R = L/(1.015L^2 + 900)

 L = poly([0],'L');
 L1 = roots(1.025*10^(6)*L^2 + 900 - 2.050*10^(6)*L^2);
 printf("\n Frame thickness = %.2f mm",2*L1(1)*10^3);
```

Scilab code Exa 7.3 Time taken to produce 1 m3 of filtrate and pressure in this time

```
clear;
clc;
printf("\n Example 7.3");
V = 0.094;          //volume in m^3
deltaP = -3530;     //P is in kN/m^2

//At t = 1105 secs
 V1 = 0.166;        //V is in m^3
 deltaP1 = -5890;    //P is in kN/m^2

a = [2.21*10^(6) -0.094;6.51*10^(6) -0.166];
b = [0.0088;0.0276];
x = inv(a)*b;
y = [x(2);x(1)];
printf("\n LA/v =%f       A^2/ r  v  = %f*10^(-7)",y(1),y(2)*10^7);
printf("\n For the full size plant:");
printf("\n LA/v = %f       A^2/ r  v =%f*10^(-7)",10*y(1),y(2)*10^8);

```

```
//Solving LHS of the integral
LHS = integrate('b+0.154+2.31','b',0,1);
//Equating LHS = RHS
t = LHS/(3.46*10^(-3));
printf("\n t = %d secs",t);
printf("\n deltaP = %dkN/m^2",(1+0.154)
    /(4.64*10^(-7)*857));
```

Scilab code Exa 7.4 Speed of rotation for maximum throughput

```
clear;
clc;
printf("\n Example 7.4");
a = [2*84300*0.02^(2)*60
    -2*0.02*0.0003;2*84300*0.02^(2)*120
    -2*0.02*0.00044];
b = [0.0003^2;0.00044^2];
x = inv(a)*b;
y = [x(2);1/x(1)];
printf("\n L/v = %f          ruv = %f*10^(10)",y(1),y
    (2)*10^(-10));
printf("\n Area of filtering surface = %f m^2",4*(
    %pi));
printf("\n Bulk volume of cake deposited =%.3f m^3/
    revolution",4*(%pi)*0.005);

V = sqrt(1*10^(-6)*143^2);
printf("\n V = %.3f m^3",V);

t =poly([0],'t');
t1 = roots(0.141^2 +2*2.19*10^(-3)
    *0.141-2*84300*(4*(%pi))^(2)*t/(3.48*10^10));
printf("\n t = %f secs",t1);
printf("\n time for 1 revolution =%.1f secs",t1/0.4)
    ;
```

```
printf("\n speed = %.3fHz",0.4/t1);
printf("\n rate of filtrate production w = %.2f kg/sec",143/67.3)
printf("\n mass of slurry S =%.1f kg/sec",1.66*2.11);
```

Scilab code Exa 7.5 Optimum filtration time for maximum throughput

```
clear;
clc;
printf("\n Example 7.5");
V1 = 0.00025;                    //V is in m^3
t = 300;                         //t is in secs
a = [7.14*10^(-6) 2.86*10^(-4);11.42*10^(-6) 2.86*10^(-4)];
b = [1.2*10^(6);1*10^(6)];
x = inv(a)*b;

//for the plate and frame filter
B1 = x(1)/(2*2.2^2*413*10^3);
B2 = x(2)/(2.2*413*1000);

printf("\nr v = %d\n",x(1));
printf("\n r l = %d",x(2));
printf("\n B1= %f    B2= %f",B1,B2);
printf("\n the filtration time for maximum throughput is:");
t1 = 21.6*10^3;
t0= t1 +B2*(t1/B1)^(0.5);
printf("\n t = %f secs",t0);
V = (t1/B1)^(0.5);
printf("\n V= %f m^3",V);
printf("\nMean rate of filtration is: %.2f *10^-6 m^3/s",(V/(t1+t0))/10^-6);
```

Scilab code Exa 7.6 Thickness of cake produced

```
clear;
clc;
printf("\n Example 7.6");
A=0.6*0.6*%pi;        //in m^2
rate=1.25*10^-4;      // in m^3/s

v_w=0.2/(3*10^3);
v_f=10^-3-v_w;

v=v_w/v_f;
v_rate=rate*v;
w=360*0.2;

t=v_rate*w/A;
printf("\nThickness of cake produced is : %.1f mm",t/10^-4);
K = poly([0],'K');
K1 = roots((1.25*10^(-4)*360)^2-K*(6.5*10^(4)*(0.36*(%pi))^(2)*72));
printf("\n The value of K is %.2f*10^(-10)",K1*10^(10));

//Filter press
//Using a filter press with n frames of thickness b m the total time, for one complete cycle of the press =(tf+120n+240),where tf is the time during which filtration is occurring
//overall rate of filtration = Vf/(tf + 120n + 240)

// Vf = 0.3^(2)*n*b/0.143
//tf = 2.064*10^5 b^2

```

```
b = poly([0],'b');
b1 = roots(b^2 - 0.0458*b - 0.001162);
printf("\n The thickness is %.4f m",b1(1));

function[n]=number_of_plates()

    n = (0.030 + 25.8*b1(1)^2)/(0.629*b1(1)-0.015);
    funcprot(0);
endfunction
n = number_of_plates();
printf("\n The minimum number of plates required is
   %d",ceil(n));

d = poly([0],'d');
d1 = roots(ceil(n)*(0.629*d-0.015)-0.030-25.8*d^2);
printf("\n The sizes of frames which will give
    exactly the required rate of filtration when six
    are used are %f mm",d1*10^3);
printf("\n\n\n  Thus any frame thickness between 47
   and 99 mm will be satisfactory. In practice,50 mm
    (2 in) frames would probably be used.")
```

Scilab code Exa 7.7 Increase in the overall throughput of the press

```
clear;
clc;
printf("\n Example 7.7");
//Case 1

                     //dV/dt = A^2(-deltaP)/vru(V + AL/v
                        ) = a/V+b
                     //For constant rate filtration:
                     //Vo/to = a/Vo + b
                     //Vo^2 + bVo = ato
                     //For constant pressure filtration
```

```
                    //0.5(V^2 - Vo^2)+b(V-Vo)=a(t-to)
                    //to=600s,t-to=3600s,Vo=V/4
                    //V^2/16 +bV/4 = 600a
                    //o.5(V^2 - V^2/16)+b(V-V/4)=3600a
                    //3600a = (15/32)V^2 +3/4(bV) = 3/8(V^2) + 3/2(bV)
                    //b = V/8
                   // a = (V^2/16 + V^2/32)/600 = (3/19200)V^2
                   //Total cycle time = 900 + 4200 = 5100secs
                   //Filtration rate = V/5100 = 0.000196V

//Case 2
                   //V1/t1 = a/ (V1 +b/4)=Vo/to=a/(Vo+b)
                   //0.5*(49/64V^2 - V1^2)+b/4(7/8V-V1)=a(t-t1)
                   //V/2400 = (3/19200)V^2/(V1+V/32)
                   //t1 = (to/Vo)V1
  t1 = 600/(1/4)*(11/32);
  printf("\n t1 = %dsecs",t1);
  //Substituting gives
  deltaT = (19200/3)*(784-121+34)/2048;
  printf("\n t -t1 = %d secs",deltaT);
  Cycle_time = 180+900+t1+deltaT;
  printf("\n cycle time = %d secs",Cycle_time);
  Increase = (0.000214 - 0.000196)/(0.000196)*100;
  printf("\n Increase in filtration rate is %.1f per cent",Increase);
```

Chapter 8

Membrane Seperation Processes

Scilab code Exa 8.2 Area of membrane and average flux

```
clear;
clc;
printf("\n Example 8.2");

//From the gel polarisation model:
         //J = (1/A)(dV/dt) = hD ln(Cg/Cf)
         //Cf = Co(Vo/V)
  //where Co and Vo are the initial concentration and volume,respectively and Cf and V are the values at subsequent times
  //Combining these eq gives
        //dV/dt = A(hDln(Cg/Co)-hDln(Vo/V))
V = [10 5 3 2 1];
y = [9.90 13.64 18.92 27.30 112.40];
plot(V,y)
xtitle("Area under the curve is 184.4","Volume(m^3)","(J - hDln(Vo/V))^(-1)")


```

```
//(b)
Jo = 0.04*log(250/20);
printf("\n Jo = %.3f m/h",Jo);
Jf = 0.04*log(250/200);
printf("\n Jf = %f m/h",Jf);
Jav = Jf + 0.27*(0.101-0.008);
printf("\n Jav = %f m/h",Jav);
//For the removal of 9m^3 filtrate in 4 hours
Area = (9/4)/Jav;
printf("\n Area = %fm^2",Area);
```

Scilab code Exa 8.3 Minimum number of membrane modules required

```
clear;
clc;
printf("\n Example 8.3");
//It is assumed that Q0 is the volumetric flowrate
   of feed
// Q2 the volumetric flowrate of concentrate
//C0 the solute concentration in the feed
// C2 the solute concentration in the concentrate
// F the volumetric flowrate of membrane permeate
// A the required membrane area.
// It is also assumed that there is no loss of
   solute through the membrane.
Cl = 3;
while 1
    Clnew = Cl -(0.04-0.02*log(30/Cl))/(Cl^(-1)/50);
    if Clnew == Cl then
        break;
    end
    Cl = Clnew;
end
    printf("\n Cl = %d kg/m^3",Cl);
printf("\n below this concentration the membrane
```

```
flux is 0.04 m/h");

//This does not pose a constraint for the single stage as the concentration of solute C2 will be that of the final concentrate, 20 kg/m3.
//Conservation of solute gives:QoCo = Q2C2
//A fluid balance gives : Qo = F + Q2
//Combining these eq and substituting Known values:
A = (2.438/0.02)/log(30/20);
printf("\n A = %d m^2",A);
//The tubular membranes to be used are available as 30 m^2modules.
printf("\n the no of required modules are %d ",A/30);

//Part(b)

//Conservation of solute gives = QoCo = Q1C1 = Q2C2
//A fluid balance on stage 2 gives Q1 = Q2 + F2
//A fluid balance on stage 2 gives Q1 = Q2 +F2
//Substituting given values in above eqns
//2.5 = 1.25/C1 + 0.02A1ln(30/C1)
function[A1]=a(C1)
    A1 = (2.5-1.25/C1)/(0.02*log(30/C1));
    funcprot(0);
endfunction
function[A2]=b(C1)
    A2 = (1.25/C1 - 0.0625)/0.00811
    funcprot(0);
endfunction

printf("\n The procedure is to use trial and error to estimate the value of C1 that gives the optimum values of A1 and A2");
printf("\n If C1 = 5kg/m^3 then A1 = %d m^2 and A2 = %d m^2",a(5),b(5));
printf("\n an arrangement of 3 modules   1 module is required.");
```

```
printf("\n\n\n   If C1 = 4 kg/m^3 then A1 = %dm^2 and
     A2 = %dm^2",a(4),b(4));
printf("\n an arrangement of 2 modules   1 module
    is almost sufficient.");
printf("\n\n\n    If C1 = 4.5 kg/m^3 then A1 = %dm^2
    and A2 = %d m^2",a(4.5),b(4.5));
printf("\n an arrangement of 2 modules   1 module
    which meets the requirement");
printf("\n\n This arrangement requires the minimum
    number of modules.");
```

Chapter 9

Centrifugal Seperations

Scilab code Exa 9.1 Value of capacity factor

```
clear;
clc;
printf("\n Example 9.1");
d_particle = 5;          //particle size is in um
p = 1000;               //density of water in kg/m^3
ps = 2800;              //density of solids in kg/m^3
viscosity = 10^(-3);    //viscosity is in Ns/m^2
uo = ((d_particle*10^(-6))^2)*(ps-p)*9.81/(18*viscosity);
printf("\n Terminal falling velocity of particles of diameter = %.2f m/sec",uo*10^5);
Q = 0.25;              //volumetric flow rate is in m^3/sec
printf("\n E = %.2f*10^(4) m^2",(Q/uo)*10^(-4));

printf("\n For coal-in-oil mixture");
uo1 = 0.04/(Q/uo);
printf("\n uo = %.2f*10^-6 m/sec",uo1*10^6);

d = sqrt((18*0.01*uo1)/((1300-850)*9.81));
```

```
printf("\n d = %d um",(d/3)*10^6);
```

Scilab code Exa 9.3 TIme taken to produce filtrate

```
clear;
clc;
printf("\n Example 9.3");
//In the filter press
// V^2 + 2(AL/v)V = 2(-deltaP)A^2*t/(ruv)

l = 0.025;          //l is in meters
L = 0.003;          //L is in meters
deltaP = 350;       //it is in N/m^2
t = 3600;           //t is in secs

// x = v/ru
x = poly([0],'x');
x1 = roots(0.025^2 + 2*0.003*0.025 - 2*3.5*10^(5)
    *3600*x);
printf("\n the value of ru/v = %.2f*10^12",(1/x1)
    *10^(-12));

//In the centrifuge
R = 0.15;               //R is in meters
H = 0.20;               //H is in meters
V = 0.00225;            //V is in m^3
r = poly([0],'r');
r1 = roots(%pi*(R^2 - r^2)*H-V);
printf("\n Value of ro = %f mm",r1(1)/2);
printf("\n angular frequency = %.1frad/s",(r1(1)
    /2*10^3)*2*(%pi));

//(R^2 -r^2)(1+2L/R)+2r^2ln(r/R) = 2vtpw^2/ru(R^2-ro
    ^2)
t = poly([0],'t');
```

```
t1 = roots((R^2 - r1(1)^2)*(1+2*(L/R))+2*(r1(1)^2)*
    log(r1(1)/R)-2*t*1000*408.4^(2)/(3.25*10^12)*(R
    ^2-(r1(1)/2)^2));
printf(" \n time required = %f secs",t1);
```

Chapter 10

Leaching

Scilab code Exa 10.1 Time required for solute to dissolve

```
clear
clc;
printf("\n Example 10.1");
//For the pilot scale vessel
 c = (2.5*75)/100;         //in kg/m^3
 cs = 2.5;                 //in kg/m^3
 V = 1.0;                  //V is in m^3
 t = 10;                   //t is in secs

 //1.875 = 2.5(1-e^(-kA/b*100))
 // x =kA/b
 x = -log(1-1.875/2.5)/10;

 //For the full scale vessel
 c = (500*28/100)/100;
 printf("\n C = %fkg/m^3",c);
 cs = 2.5;                  //cs is in kg/m^3
 V = 100;                   //V is in m^3
 t = -log(1-1.4/2.5)*(100/0.139);  //t is in secs
 printf("\n t = %d secs",t);
```

Scilab code Exa 10.2 Rate of feed of neutral water to the thickeners

```
clear;
clc;
printf("\n Example 10.2");

//If x1, x2, x3 are the solute: solvent ratios in
   thickeners 1, 2, and 3, respectively, the
   quantities of CaCO3. NaOH, and water in each of
   the streams can be calculated for every 100 kg of
    calcium carbonate

//Since the final underflow must contain only 1 per
   cent of NaOH
function[f]=F(x)
    f(1)=(300*x(3))/100 - 0.01;
    f(2)=300*(x(2)-x(3))/x(4) - x(3);         //Wf =
       x(4)
    f(3)=300*(x(1)-x(3))/x(4) - x(2);
    f(4)=(80-300*x(3))/(600+x(4))-x(1);
    funcprot(0);
endfunction
//An initial guess
x = [0.1 0.1 0.1 0.1];
y = fsolve(x,F);
printf("\n x1 = %f  x2 = %f  x3 = %f Wf = %f",y(1),y
   (2),y(3),y(4));

printf("\n Thus the amount of water required for
   washing 100 kg CaCO3 is %f kg",y(4));
printf("\n The solution fed to reactor contains 0.25
    kg/s Na2CO3. This isequivalent to 0.236 kg/s
   CaCO3,and hence the actual water required is %.2f
    kg/sec",y(4)*0.236/100);
```

Scilab code Exa 10.3 Required number of thickners

```
clear;
clc;
printf("\n Example 10.3");

//part 1
//Solvent in underflow from final washing thickener = 50 kg/s.
//The solvent in the overflow will be the same as that supplied for washing (200 kg/s).
//Solvent discharged in overflow/Solvent discharged in underflow=  forthe washing thickeners.
//Liquid product from plant contains 54.9 kg of salt in 195 kg of solvent.
//This ratio will be the same in the underflow from the first thickener.
printf("\n the material fed to the washing thickeners consists of 100 kg TiO2, 50 kg solvent and %d kg salt",50*(54.9/195));

//m =n+1
m = log(421)/log(4);
printf("\n The required number of thickeners for washing are %f",m);

//Part 2
//From an inspection of the data, it is seen that Wh = 0.30 + 0.2Xh.
//Thus: Sh = WhXh = 0.30Xh + 0.2X2h= 5Wh^2     1.5Wh
//Considering the passage of unit quantity of TiO2 through the plant, then:
Ln = 0; wn = 2; Xn = 0;
//since 200 kg/s of pure solvent is used.
```

```
Sn = 0.001;
Wn = 0.3007;
So = 0.55;
Wo = 1.00;
//X1 = (Ln+1 + S0    Sn)/(wn+1 + W0    Wn)
X1 = (0+0.55-0.001)/(2+1-0.3007);
printf("\n concentration in the first thickener is
   %f",X1);

W1 = 0.30+0.2*0.203;
printf("\n W1 = %f",W1);
S1 = (0.3406*0.203);
printf("\n S1 = %f",S1);
X2 = (0.0691 - 0.001)/(1.7 + 0.3406);
printf("\n X2 = %f",X2);
W2 = (0.30+0.2*0.0334);
printf("\n W2 = %f",W2);
X3 = (0.01025-0.001)/(1.7+0.3067);
printf("\n X3 = %f",X3);
printf("\n W3 = 0.30089");
printf("\n S3 = %f",0.0013);
printf("\n W4 = %f and S4 = %f",0.30003,0.000045);
printf("\n Thus S4 is less than Sn and therefore 4
   thickeners are required.");
```

Scilab code Exa 10.4 Number of ideal stages required

```
clear;
clc;
printf("\n Example 10.4");
//Since the seeds contain 20 per cent of oil
xAo=0.2;
xBo=0.8;
printf("\n xAo = %.1f and xBo = %.1f",xAo,xBo);
//The final solution contains 50 per cent oil
```

```
yA1=0.5;
yS1=0.5;
printf("\n yA1 = %.1f and yS1 = %.1f",yA1,yS1);
//The solvent which is used for extraction is pure
    and hence
ySn1=1;
//1 kg of insoluble solid in the washed product is
    associated with 0.5 kg of solution and 0.025 kg
    oil.
xAn = 0.0167;
xBn = 0.6667;
xSn = 0.3166;

//The mass fraction of insoluble material in the
    underflow is constant and equal to 0.667. The
    composition of the underflow is therefore
    represented, on the diagram Figure 10.22, by a
    straight line parallel to the hypotenuse of the
    triangle with an intercept of 0.333 on the two
    main axes.

//The difference point is now found by drawing in
    the two lines connecting x0 and y1 and xn and yn
    +1.

//The graphical construction described in the text
    is then used and it is seen from Figure 10.22
    that xn lies in between x4 and x5.
printf("\n Thus 5 thickeners are adequate and for
    the required degree of extraction");
```

Scilab code Exa 10.5 Number of theoretical stages required

```
clear;
clc;
```

```
printf("\n Example 10.5");

//On the basis of 100 kg untreated solids
//In the underflow feed:
//0.35 kg oil is associated with each kg of exhausted livers.
printf("\n mass of livers fed=%d kg contaning %d kg oil",100/(1+0.35),100-74);
xA = 0.26;
printf("\n xA = %.2f",xA);
xs = 0;
//In the overflow feed, pure ether is used
ys =1.0;
xs = 0;
//Recovery = 90 per cent
printf("\n                         Exhausted livers          Oil           Ether");
printf("\n Underflow feed              %d              %d              -",74,26);
printf("\n Overflow feed               -               -              %d",50);
printf("\n Underflow product           %d              %d              e(say)",74,2.6);
printf("\n Overflow product            -               %d              50-e",23.4);
printf("\n In the underflow product:");
printf("\n the ratio(oil/exhausted livers) = %.3f kg/kg",2.6/74);
printf("\n Ratio(ether/exhausted livers) = %.3f kg/kg",0.306);
printf("\n e = %.1f kg",0.306*74);
//In the overflow product:
printf("\n The mass of ether = %.1f kg",50-22.6);
printf("\n yA = %.2f",23.4/(23.4+27.4));
printf("\n ys = %.2f",1-23.4/(23.4+27.4));
printf("\n ");
```

Chapter 11

Distillation

Scilab code Exa 11.1 Mole fraction calculation

```
//Example 11.1

clear;
clc;

printf("\tExample 11.1\n");

p0_A=106;  //Vapour pressure of n-heptane  in kN/m^2
p0_B=73.7;  //Vapour pressure of toluene  in kN/m^2
P=101.3;  //Total pressure in kN/m^2

xA=(P-p0_B)/(p0_A-p0_B);

yA=p0_A*xA/P;

printf("\nMole fraction in liquid phase is : %.3f",
    xA);
printf("\nMole fraction in vapour phase is : %.3f\n"
    ,yA);

//End
```

Scilab code Exa 11.2 Saturated Pressure calculation

```
//Example 11.2

clear;
clc;

printf("\tExample 11.2\n");

Pc=4700;  //Critical pressure in kN/m^2
Tc=508.1;  //critical temperature in K
p1=100.666;  //in kN/m^2
T1=329.026;  //in K
T=350.874;  //in K

Tr1=T1/Tc;
Pr1=p1/Pc;
printf("\nTr1 = %f \nPr1 = %f\n",Tr1,Pr1);

c5=-35+(36/Tr1)+(42*log(Tr1))-(Tr1^6);
c2=((0.315*c5)-log(Pr1))/((0.0838*c5)-log(Tr1));
c1=0.0838*(3.758-c2);
printf("\nc5 = %.4f \nc2 = %.4f \nc1 = %.4f\n",c5,c2,c1);

k9=-35*c1;
k10=-36*c1;
k11=(42*c1)+c2;
k12=-c1;

printf("\nk9 = %.3f \nk10 = %.3f \nk11 = %.4f",k9,k10,k11);
printf("\nk12 = %.5f\n",k12);

```

```
Tr=T/Tc;
Pr=exp(k9-(k10/Tr)+(k11*log(Tr))+(k12*Tr^6));
p0=Pc*Pr;
printf("\nPr = %f \n\nP0 = %.2f kN/m^2\n",Pr,p0);

//End
```

Scilab code Exa 11.3 Vapour phase composition of a mixture

```
//Exaple  11.3

clear;
clc;

printf("\tExample 11.3\n");

k1_B=6.90565;
k2_B=1211.033;
k3_B=220.79;

k1_T=6.95334;
k2_T=1343.943;
k3_T=219.377;

t=338-273;  //in Degree celsius
P=101.3;  //in kN/m^2
xB=0.5;
xT=0.5;

function[p0]=antoine(k1,k2,k3,T)
    p0=10^(k1-(k2/(T+k3)));
    funcprot(0)
endfunction

p0_B=antoine(k1_B,k2_B,k3_B,t)*101.325/760;
```

```
p0_T=antoine(k1_T,k2_T,k3_T,t)*101.325/760;

printf("\nP0_B = %.1f kN/m^2\nP0_T = %.1f kN/m^2\n",
    p0_B,p0_T);

pB=xB*p0_B;
pT=xT*p0_T;
printf("\npB = %.2f kN/m^2 \npT = %.3f\n",pB,pT);

p=pB+pT;
yB=pB/p;
yT=pT/p;
printf("\nyB = %.3f \nyT = %.3f\n",yB,yT);

//End
```

Scilab code Exa 11.4 Boiling point of equimolar mixture

```
//Example 11.4

clear;
clc;

printf("\tExample 11.4\n");

function[p0]=antoine(k1,k2,k3,T)
    p0=10^(k1-(k2/(T+k3-273)));
    funcprot(0)
endfunction

k1_B=6.90565;
k2_B=1211.033;
k3_B=220.79;

k1_T=6.95334;
```

```
k2_T=1343.943;
k3_T=219.377;

xB=0.5;
xT=0.5;

printf("\n\tT(K)");
T=[373 353 363 365 365.1];
disp(T);

i=1;

while i<6
    p0_B(i)=antoine(k1_B,k2_B,k3_B,T(i))*101.325/760;
    p0_T(i)=antoine(k1_T,k2_T,k3_T,T(i))*101.325/760;
    pB(i)=xB*p0_B(i);
    pT(i)=xT*p0_T(i);
    p(i)=pB(i)+pT(i);
    i=i+1;
end
printf("\n\tp0 B")
disp(p0_B);
printf("\n\tp0 T");
disp(p0_T);
printf("\n\tpB");
disp(pB);
printf("\n\tpT");
disp(pT);
printf("\n\tpB+pT");
disp(p);

//since total pressure at 365.1 K is nearly same as 101.3 kPa
printf("\nBoiling temperature is %f K",T(5));

//End
```

Scilab code Exa 11.5 Dew point of a equimolar mixture

```
//Example 11.5

clear;
clc;

printf("\tExample 11.5\n");

function[p0]=antoine(k1,k2,k3,T)
    p0=10^(k1-(k2/(T+k3-273)));
    funcprot(0)
endfunction

k1_B=6.90565;
k2_B=1211.033;
k3_B=220.79;

k1_T=6.95334;
k2_T=1343.943;
k3_T=219.377;

//Since total pressure is 101.3 kPa, pB=pT
pB=50.65;
pT=50.65;

printf("\n\tT(K)");
T=[373.2 371.2 371.7 371.9 372];
disp(T);

i=1;

while i<6
    p0_B(i)=antoine(k1_B,k2_B,k3_B,T(i))
```

```
        *101.325/760;
    p0_T(i)=antoine(k1_T,k2_T,k3_T,T(i))
        *101.325/760;
    xB(i)=pB/p0_B(i);
    xT(i)=pT/p0_T(i);
    x(i)=xB(i)+xT(i);
    i=i+1;
end

printf("\n\tp0 B")
disp(p0_B);
printf("\n\tp0 T");
disp(p0_T);
printf("\n\txB");
disp(xB);
printf("\n\txT");
disp(xT);
printf("\n\tx = xB+xT");
disp(x);

//Since last value is closer to 1, 372 K is the
    required dew point
printf("\nDew point can be taken as %f K",T(5));

//End
```

Scilab code Exa 11.6 Composition of vapour and liquid

```
//Example 11.6

clear;
clc;

printf("\tExample 11.6\n");

```

```
//Fractional vapourisation   f=V/F
f=0.25;

slope=(1-f)/-f;

//from fig 11.9, a construction is made
printf("\nFor slope = %d",slope);
printf("\nx=0.42 \t y=0.63\n");

//from fig. 11.10,temperature corresponding to x
   =0.42 is noted
printf("\nfor x=0.42 \t T=366.5");

//End
```

Scilab code Exa 11.7 Number of theoretical plates needed

```
clear;
clc;
printf('Example 11.7'); //Example 11.7
// Find Number of theoretical plates needed and the
   position of entry for the feed

F = 100;                //Feed [kmol]

function[f]=Feed(x)
    f(1)=x(1)+x(2)-100;                    //Overall
        mass Balance
    f(2)=0.9*x(1)+.1*x(2)-(100*.4);        //A
       balance on MVC, benzene
    funcprot(0)
endfunction
x = [50 50];
product = fsolve(x,Feed);

```

```
//Using notation of figure 11.13
Ln = 3*product(1);
Vn = Ln + product(1);

//Reflux to the plate
Lm = Ln + F;
Vm = Lm - product(2);

//Equilibrium Composition
xt = .79;    yt = .9;
//From Top eqm line
yt1 = (Ln/Vn)*xt + (product(1)/Vn);
xt1=.644;          //Thus from Eqm curve for yt1
//From Top eqm line
yt2 = (Ln/Vn)*xt1 + (product(1)/Vn);
xt2=.492;          //Thus from Eqm curve for yt2
//From Top eqm line
yt3 = (Ln/Vn)*xt2 + (product(1)/Vn);
xt3=.382;          //Thus from Eqm curve for yt3
//From II Eqm Line
yt4 = (Lm/Vm)*xt3 - (product(2)/Vm)*.1;
xt4=.2982;          //Thus from Eqm curve for yt4
//From II Eqm Line
yt5 = (Lm/Vm)*xt4 - (product(2)/Vm)*.1;
xt5=.208;          //Thus from Eqm curve for yt5
//From II Eqm Line
yt6 = (Lm/Vm)*xt5 - (product(2)/Vm)*.1;
xt6=.120;          //Thus from Eqm curve for yt6
//From II Eqm Line
yt7 = (Lm/Vm)*xt6 - (product(2)/Vm)*.1;
xt7=.048;          //Thus from Eqm curve for yt7

//Equilibrium Data
y=[0 yt7 yt6 yt5 yt4 yt3 yt2 yt1 yt];
x=[0 xt7 xt6 xt5 xt4 xt3 xt2 xt1 xt];
//Top Equilibrium Line equation 11.35
x1 = linspace(0,.79,100);
y1 = (Ln/Vn)*x1 + (product(1)/Vn);
```

```
//Equilibrium Line equation 11.37
x2 = linspace(0.048,.44,100);
y2 = (Lm/Vm)*x2 - (product(2)/Vm)*.1;
clf();
plot(x,y,x1,y1,x2,y2);
xtitle("Lewis-Sorel Method", "Mole fraction of C6H6 in Liquid (x)", "Mole Fraction C6H6 in Vapor (y)");
legend ("Equilirium Plot", "Top Eqm Line", "Bottom Eqm Line");
printf("\n\n As the least point on equilibrium Line xt-7 correspond to reboiler , and there will be seven plates");


//END
```

Scilab code Exa 11.8 The Mc Cabe Thiele method

```
clear;
clc;
printf('Example 11.8'); //Example 11.8
// Find Number of theoretical plates needed and the position of entry for the feed by mccabe thiele method

F = 100;                    //Feed [kmol]

function[f]=Feed(x)
    f(1)=x(1)+x(2)-100;                         //Overall mass Balance
    f(2)=0.9*x(1)+.1*x(2)-(100*.4);             //A balance on MVC, benzene
    funcprot(0)
endfunction
```

```
x = [50 50];
product = fsolve(x,Feed);

//Using notation of figure 11.13
Ln = 3*product(1);
Vn = Ln + product(1);

//Reflux to the plate
Lm = Ln + F;
Vm = Lm - product(2);

//Equilibrium Data
y=[0 .127 .252 .379 .498 .594 .708 .818 .9 1];
x=[0 .048 .12 .208 .298 .382 .492 .644 .79 1];
//Diagnol Line
y3 = [0 1];
x3 = [0 1];
//Top Equilibrium Line equation 11.35
x1 = linspace(0,.985,100);
y1 = (Ln/Vn)*x1 + (product(1)/Vn);
//Equilibrium Line equation 11.37
x2 = linspace(0.048,.44,100);
y2 = (Lm/Vm)*x2 - (product(2)/Vm)*.1;
clf();
//Setting initial point A x = .985 at top eqm line
xm = [.985 .965 .965 .92 .92 .825 .825 .655 .655 .44
    .44 .255 .255 .125 .125 .048];
ym = [.985 .985 .965 .965 .92 .92 .825 .825 .655
    .655 .44 .44 .255 .255 .125 .125];
xp = [.985 .965 .92 .825 .655 .44 .255 .125 .048];
yp = [.985 .965 .92 .825 .655 .44 .255 .125 .048];
plot(x,y,x3,y3,x1,y1,x2,y2,xm,ym);
xtitle("Mccabe Thiele Method", "Mole fraction of
   C6H6 in Liwuid (x)", "Mole Fraction C6H6 in Vapor
    (y)");
legend ("Equilirium Plot", "Diagnol Line","Top Eqm
   Line", "Bottom Eqm Line",5);
xset('window',1);
```

```
for(i=2:8)
plot(xp(i),yp(i),"o-");
xtitle("Equilibrium plot","mole fraction C6H6 in liquid(x)","mole fractionC6H6 in vapour(y)");
end
printf("\n\n The Number of stages are then counted highlighted points that is number of plates required as 7");


//END
```

Scilab code Exa 11.9 Number of plates required at total reflux

```
//Example 11.9

clear;
clc;

printf("\tExample 11.9\n");

xA_d=0.9;
xA_s=0.1;
xB_d=1-xA_d;
xB_s=1-xA_s;

a=2.4;
xd=0.9;
xf=0.4;
xw=0.1;

n=(log(xA_d*xB_s/(xB_d*xA_s))/log(a))-1;
printf("\nNo. of theoretical plates in column is : %d",n);

```

```
Rm=((xd/xf)-(a*((1-xd)/(1-xf))))/(a-1);
printf("\nMinimum reflux ratio is %.2f\n",Rm);

yf=0.61;
printf("\nUsing Graphical construction with yf=0.61\
   n");
Rmin=(xd-yf)/(yf-xf);
printf("Minimum reflux ratio is %.2f\n",Rmin);

//End
```

Scilab code Exa 11.10 Heat required

```
//Example 11.10

clear;
clc;

printf("\tExample 11.10\n");

//From material balance
    //  D+W=1
    //  0.995D+0.1W=1*3

A=[1 1;0.995 0.1];
B=[1;3];
Rm = (1952-1547)/(1547-295);
printf("\n Rm = %.3f",Rm);
NA = 1.08*405;
printf("\n Since the actual reflux is 8 pre cent
   above the minimum NA = 1.08*NmA = %.3f",NA);
N = 5/0.6;
printf("\n Number of plates to be  required are %.3f
   ",5/0.6);

```

```
Qb_W = 582 - (-209);
printf("\n Heat input to the boiler per unit mass of
    bottom product is %.3f",Qb_W);
printf("\n Heat input to the boiler = %.3f kW"
   ,791*0.78);
printf("\n Condenser duty = %d kW",(1984-296)*0.22);
```

Scilab code Exa 11.11 Number of theoretical stages required

```
//Example 11.11

clear;
clc;

printf("\tExample 11.11\n");

//    F is feed
//    D is distillate
//    W is waste
//    S is sidestream

F=100;
S=10;

//Mass fractions of CCl4 in various streams
xf=0.5;
xd=0.95;
xw=0.05;
xs=0.8;

//    D + W = 100-10
//    0.95D + 0.00W = 50-8
A=[1,1;0.95,0.05];
B=[90;42];
x=inv(A)*B;
```

```
printf("\nD = %.1f     W = %.1f\n",x(1),x(2));

disp("From the enthalpy data and the reflux ratio, the upper pole point M is located as shown in Figure.");

disp("Points F and S are located,such that FS/FF = 10.");

disp("MF is Joined and extended to cut NS^A at O, the immediate pole point.");
disp("The number of stages required is then obtained from the figure and");
printf("13 theoretical stages are required");

//End
```

Scilab code Exa 11.12 Amount of distillate

```
clear;
clc;
printf("\n Example 11.12");
R = [0.85 1.0 1.5 2.0 3.0 4.0];     //Reflux ratio
xd = 0.75;                           //top concentration of alcohol
xs = [0.55 0.50 0.37 0.20 0.075 0.05];//From the graph fig.11.35 page-596
Db(1) = 0;

printf("\n    R                Fi                    xs                Db    ");
i=1;
while i<=6
    Fi(i) = xd/(R(i) + 1);
    if i>1 then
```

```
    Db(i) = 100*(xs(1)-xs(i))/(xd-xs(i));
    end
   printf("\n %.2f          %.3f             %.2f          %.1f",R(i),Fi(i),xs(i),Db(i));
    i=i+1;
end
plot(R,Db);
xtitle("","Reflux ratio(R)","Product Db (kmol)");
printf("\n The area under the Db vs R curve is given by 96 kmol");
Hav = 4000;              //average latent heat in kJ/kmol
Qr = 96*Hav/1000;
printf("\n Heat to be supplied to provide the reflux,Qr is approximately %.1f MJ",Qr);
printf("\n Heat to be supplied to provide the reflux per kmol of product is then %.2f MJ",380/71.4);
printf("\n Total heat = %.2f MJ/kmol product",5.32+4.0);
```

Scilab code Exa 11.13 Heat required and average composition

```
//Example 11.13

clear;
clc;

printf("\tExample 11.13\n");

xs=[0.55;0.5;0.425;0.31;0.225;0.105];
xd=[0.78;0.775;0.77;0.76;0.75;0.74];
differ=xd-xs;
for i=1:6
    reci(i)=1/(xd(i)-xs(i));
end
```

```
m=[xs xd differ reci];
printf("\n      xs        xd        (xd-xs)    1/(xd-xs)\n
  ");
disp(m);
plot(xs,reci);
xtitle('Graphical integration','xs','1/xd-xs)');

//Area under the curve is calculated and is found to
    be 1.1

//logdiv = S1/S2 = area under the curve
logdiv=1.1;
S1=100;              //Assume
div=exp(logdiv);
S2=S1/div;
Db=S1-S2;          //Product obtained
amt=xs(1)*S1-(xs(6)*S2);
avg=amt/Db;

printf("\nAverage composition is %.2f kmol\n",avg);

L=4000;            //latent heat
R=2.1;

h=R*L;
printf("Heat required to produce reflux per kmol :
  %d kJ\n",h);

//End
```

Scilab code Exa 11.14 Ideal plates required

```
clear;
clc;
```

```
xdo = 0.98;                  //per cent of ortho top product
xwo = 0.125;                 //per cent of ortho bottom product

function[f]=product(x)
    f(1) = 100 - x(1) - x(2);       //x(1) is D and x(2) is W
    f(2) = 60 - x(1)*xdo - x(2)*xwo;
    funcprot(0);
endfunction
x = [0,0];
y = fsolve(x,product)
printf("\n D = %.2f kmol & W = %.2f kmol",y(1),y(2));

printf("\n Let us assume that the distillate contains 0.6 mole per cent meta and 1.4 mole per cent para");
printf("\n Component         Feed                    Distillate                    Bottoms          ");
printf("\n              (kmol)   (mole per cent)      (kmol)  (mole per cent)   (kmol)    (mole per cent)  ");
printf("\n Ortho  %.3f       %.2f                 %.2f            %.2f                        %.2f            %.2f            ",60,60,y(1)*0.98,98,y(2)*0.125,12.5);
printf("\n Meta   %.3f             %.2f                        %.2f          %.2f                             %.2f              %.2f             ",4,4,y(1)*0.006,0.6,y(2)*0.083,8.3);
printf("\n Para   %.3f       %.2f                       %.2f        %.2f                    %.2f            %.2f          ",36,36,y(1)*0.014,1.4,y(2)*0.792,79.2);

ao = 1.7;               //relative volatility of ortho relative to para
```

```
am = 1.16;              //relative volatility of meta relative to para
ap =1;                  //relative volatility of para w.r.t. to itself
xso = 0.125;
xsm = 0.083;
xsp = 0.792;
xwo = 0.125;
xwp = 0.083;
xwm = 0.792;
yso = ao*xso/(ao*xso+ap*xsp+am*xsm);
ysm = am*xsm/(ao*xso+ap*xsp+am*xsm);
ysp = ap*xsp/(ao*xso+ap*xsp+am*xsm);
//Equations of operating lines
//Above the feed point
Ln = 5*y(1);            //Liquid downflow
Vn = 6*y(1);            //Vapour up
//Assuming the feed is liquid at its boiling point
F = 100;            //feed
Lm = Ln+F;          //liquid downflow
Vm = Lm-y(2);       //Vapour up
x1o = poly([0],'x1o');
x11 = roots(yso - (Lm/Vm)*x1o + (y(2)/Vm)*xwo);
x1p = poly([0],'x1p');
x12 = roots(ysp - (Lm/Vm)*x1p + (y(2)/Vm)*xwp);
x1m = poly([0],'x1m');
x13 = roots(ysm - (Lm/Vm)*x1m + (y(2)/Vm)*xwm);
x1 = [x11 x13 x12];
ax1 = [ao*x11 am*x13 ap*x12];
y1 = [ax1(1)/(ax1(1)+ax1(2)+ax1(3)) ax1(2)/(ax1(1)+ax1(2)+ax1(3)) ax1(3)/(ax1(1)+ax1(2)+ax1(3))];
x2o = poly([0],'x2o');
x21 = roots(y1(1) - (Lm/Vm)*x2o + (y(2)/Vm)*xwo);
x2p = poly([0],'x2p');
x22 = roots(y1(3) - (Lm/Vm)*x2p + (y(2)/Vm)*xwp);
x2m = poly([0],'x2m');
x23 = roots(y1(2) - (Lm/Vm)*x2m + (y(2)/Vm)*xwm);
x2 = [x21 x23 x22];
```

```
printf("\n              plate  compositions  below  the feed  plate");
printf("\n Component                  xs                axs                  ys                 x1                  ax1                 y1                 x2");
printf("\n      o                    %.3f            %.3f              %.3f          %.3f             %.3f          %.3f          %.3f",xso,ao*xso,yso,x1(1),ax1(1),y1(1),x2(1));
printf("\n      m                    %.3f            %.3f              %.3f          %.3f             %.3f          %.3f          %.3f",xsm,am*xsm,ysm,x1(2),ax1(2),y1(2),x2(2));
printf("\n      p                    %.3f            %.3f              %.3f          %.3f             %.3f          %.3f          %.3f",xsp,ap*xsp,ysp,x1(3),ax1(3),y1(3),x2(3));
printf("\n                           %.3f            %.3f              %.3f          %.3f             %.3f          %.3f          %.3f",xso+xsm+xsp,ao*xso+am*xsm+ap*xsp,yso+ysm+ysp,x1(1)+x1(2)+x1(3),ax1(1)+ax1(2)+ax1(3),y1(1)+y1(2)+y1(3),x2(1)+x2(2)+x2(3));

ax2 = [ao*x2(1) am*x2(2) ap*x2(3)];
y2 = [ax2(1)/(ax2(1)+ax2(2)+ax2(3)) ax2(2)/(ax2(1)+ax2(2)+ax2(3)) ax2(3)/(ax2(1)+ax2(2)+ax2(3))];
x3o = poly([0],'x3o');
x31 = roots(yso - (Lm/Vm)*x3o + (y(2)/Vm)*xwo);
x3p = poly([0],'x3p');
x32 = roots(ysp - (Lm/Vm)*x3p + (y(2)/Vm)*xwp);
x3m = poly([0],'x3m');
x33 = roots(ysm - (Lm/Vm)*x3m + (y(2)/Vm)*xwm);
x3 = [x31 x33 x32];

ax3 = [ao*x3(1) am*x3(2) ap*x3(3)];
y3 = [ax3(1)/(ax3(1)+ax3(2)+ax3(3)) ax3(2)/(ax3(1)+ax3(2)+ax3(3)) ax3(3)/(ax3(1)+ax3(2)+ax3(3))];
```

```
x4o = poly([0],'x4o');
x41 = roots(yso - (Lm/Vm)*x4o + (y(2)/Vm)*xwo);
x4p = poly([0],'x4p');
x42 = roots(ysp - (Lm/Vm)*x4p + (y(2)/Vm)*xwp);
x4m = poly([0],'x4m');
x43 = roots(ysm - (Lm/Vm)*x4m + (y(2)/Vm)*xwm);
x4 = [x41 x43 x42];

ax4 = [ao*x4(1) am*x4(2) ap*x4(3)];
y4 = [ax4(1)/(ax4(1)+ax4(2)+ax4(3)) ax4(2)/(ax4(1)+
    ax4(2)+ax4(3)) ax4(3)/(ax4(1)+ax4(2)+ax4(3))];
x5o = poly([0],'x5o');
x51 = roots(yso - (Lm/Vm)*x5o + (y(2)/Vm)*xwo);
x5p = poly([0],'x5p');
x52 = roots(ysp - (Lm/Vm)*x5p + (y(2)/Vm)*xwp);
x5m = poly([0],'x5m');
x53 = roots(ysm - (Lm/Vm)*x5m + (y(2)/Vm)*xwm);
x5 = [x51 x53 x52];

ax5 = [ao*x5(1) am*x5(2) ap*x5(3)];
y5 = [ax5(1)/(ax5(1)+ax5(2)+ax5(3)) ax5(2)/(ax5(1)+
    ax5(2)+ax5(3)) ax5(3)/(ax5(1)+ax5(2)+ax5(3))];
x6o = poly([0],'x6o');
x61 = roots(yso - (Lm/Vm)*x6o + (y(2)/Vm)*xwo);
x6p = poly([0],'x6p');
x62 = roots(ysp - (Lm/Vm)*x6p + (y(2)/Vm)*xwp);
x6m = poly([0],'x6m');
x63 = roots(ysm - (Lm/Vm)*x6m + (y(2)/Vm)*xwm);
x6 = [x61 x63 x62];

ax6 = [ao*x6(1) am*x6(2) ap*x6(3)];
y6 = [ax6(1)/(ax6(1)+ax6(2)+ax6(3)) ax6(2)/(ax6(1)+
    ax6(2)+ax6(3)) ax6(3)/(ax6(1)+ax6(2)+ax6(3))];
x7o = poly([0],'x7o');
x71 = roots(yso - (Lm/Vm)*x7o + (y(2)/Vm)*xwo);
x7p = poly([0],'x7p');
x72 = roots(ysp - (Lm/Vm)*x7p + (y(2)/Vm)*xwp);
x7m = poly([0],'x7m');
```

```
x73 = roots(ysm - (Lm/Vm)*x7m + (y(2)/Vm)*xwm);
x7 = [x71 x73 x72];
printf("\n Component                ax2
   y2                  x3                  ax3
                  y3                  x4                  ax4");
printf("\n      o                          %.3f
             %.3f              %.3f          %.3f            %
   .3f         %.3f             %.3f",ax2(1),y2(1),x3(1),
   ax3(1),y3(1),x4(1),ax4(1));
printf("\n      m                          %.3f
             %.3f              %.3f          %.3f            %
   .3f         %.3f             %.3f",xsm,am*xsm,ysm,x1(2)
   ,ax1(2),y1(2),x2(2));
printf("\n      p                          %.3f
             %.3f              %.3f          %.3f            %
   .3f         %.3f             %.3f",xsp,ap*xsp,ysp,x1(3)
   ,ax1(3),y1(3),x2(3));

printf("\n Component                y4
   x5                  ax5                  y5
                  x6                  ax6                  y6");
printf("\n      o                          %.3f
             %.3f              %.3f          %.3f            %
   .3f         %.3f             %.3f",y4(1),x5(1),ax5(1),
   y5(1),x6(1),ax6(1),y6(1));
printf("\n      m                          %.3f
             %.3f              %.3f          %.3f            %
   .3f         %.3f             %.3f",y4(2),x5(2),ax5(2),
   y5(2),x6(2),ax6(2),y6(2));
printf("\n      p                          %.3f
             %.3f              %.3f          %.3f            %
   .3f         %.3f             %.3f",y4(3),x5(3),ax5(3),
   y5(3),x6(3),ax6(3),y6(3));


ax7 = [ao*x7(1) am*x7(2) ap*x7(3)];
y7 = [ax7(1)/(ax7(1)+ax7(2)+ax7(3)) ax7(2)/(ax7(1)+
   ax7(2)+ax7(3)) ax7(3)/(ax7(1)+ax7(2)+ax7(3))];
```

```
x8o = poly([0],'x8o');
x81 = roots(yso - (Ln/Vn)*x8o + (y(2)/Vn)*xwo);
x8p = poly([0],'x8p');
x82 = roots(ysp - (Ln/Vn)*x8p + (y(2)/Vn)*xwp);
x8m = poly([0],'x8m');
x83 = roots(ysm - (Ln/Vn)*x8m + (y(2)/Vn)*xwm);
x8 = [x81 x83 x82];

ax8 = [ao*x8(1) am*x8(2) ap*x8(3)];
y8 = [ax8(1)/(ax8(1)+ax8(2)+ax8(3)) ax8(2)/(ax8(1)+ax8(2)+ax8(3)) ax8(3)/(ax8(1)+ax8(2)+ax8(3))];
x9o = poly([0],'x9o');
x91 = roots(yso - (Ln/Vn)*x9o + (y(2)/Vn)*xwo);
x9p = poly([0],'x9p');
x92 = roots(ysp - (Ln/Vn)*x9p + (y(2)/Vn)*xwp);
x9m = poly([0],'x9m');
x93 = roots(ysm - (Ln/Vn)*x9m + (y(2)/Vn)*xwm);
x9 = [x91 x93 x92];

printf("\n Component              x7              ax7              y7              x8              ax8              y8              x9");
printf("\n     o              %.3f              %.3f              %.3f              %.3f              %.3f              %.3f              %.3f",x7(1),ax7(1),y7(1),x8(1),ax8(1),y8(1),x9(1));
printf("\n     m              %.3f              %.3f              %.3f              %.3f              %.3f              %.3f              %.3f",x7(2),ax7(2),y7(2),x8(2),ax8(2),y8(2),x9(2));
printf("\n     p              %.3f              %.3f              %.3f              %.3f              %.3f              %.3f              %.3f",x7(3),ax7(3),y7(3),x8(3),ax8(3),y8(3),x9(3));

printf("\n Component              x7              ax7              y7              x8              ax8              y8              x9");
```

```
printf("\n        o                              %.3f
                %.3f               %.3f          %.3f              %
    .3f            %.3f           %.3f",y4(1),x5(1),ax5(1),
    y5(1),x6(1),ax6(1),y6(1));
printf("\n        m                              %.3f
                %.3f               %.3f          %.3f              %
    .3f            %.3f           %.3f",y4(2),x5(2),ax5(2),
    y5(2),x6(2),ax6(2),y6(2));
printf("\n        p                              %.3f
                %.3f               %.3f          %.3f              %
    .3f            %.3f           %.3f",y4(3),x5(3),ax5(3),
    y5(3),x6(3),ax6(3),y6(3));
```

Scilab code Exa 11.15 Minimum reflux ratio

```
clear all;
clc;
printf("\n Example 11.15");

printf("\n Dew point calculation");
xd1 = 0.975;                //n-C4 light distillate
xd2 = 0.025;               //C5 heavy key distillate
Td = 344;                  //temperature in kelvins
K1 = 1.05;                //Eqillibrium constant
    calculation for n-C4 at 344 K
K2 = 0.41;                //Equillibrium constant
    calculation for C5 at 344K
//By a dew point calculation
//sum(xd)=sum(xd/K)
printf("\n Component        xd                    Td = 344
    K");
printf("\n                                        K
    xd/K ");
printf("\n n-C4             %.3f                  %.2f
    %.3f",xd1,K1,xd1/K1);
```

```
printf("\n C5                %.3f                %.2f
   %.3f",xd2,K2,xd2/K2);
printf("\n                   %.1f
   %.3f",xd1+xd2,(xd1/K1)+(xd2/K2));

K11 = 1.04;
K21 = 0.405;
//Calculation for xd at 343 K
x = poly([0],'x');
x1 = roots(x/K11 + (1-x)/K21);
printf("\n

   Td = 343 K");
printf("\n

   K      xd/K");
printf("\n

   %.3f    %.3f",K11,x1/K11);
printf("\n

   %.3f    %.3f",K21,(1-x1)/K21);
printf("\n

   %.3f",x1/K11+(1-x1)/K21);
printf("\n At 343 K        K1 = 1.04        K2 = 0.405
    from fig.11.39");

printf("\n\n\n Estimation of still temperature Ts");
//sum(xw) = sum(K*xw)
K31 = 3.05;           //equillibrium const at 419 K
K32 = 1.6;            //equillibrium const at 419 K
K33 = 0.87;           //equillibrium const at 419 K
K34 = 0.49;           //equillibrium const at 419 K
xw1 = 0.017;          //mass fraction of n-C4
xw2 = 0.367;          //mass fraction of C5
xw3 = 0.283;          //mass fraction of C6
xw4 = 0.333;          //mass fraction of C7
```

```

printf("\n At Ts = 416 K equillibrium constants are from fig.11.39");
printf("\n n-C4              C5            C6              C7");
printf("\n  %.2f             %.2f          %.2f            %.2f",K31,K32,K33,K34);
printf("\n\n    at Ts = 416 K");
printf("\n n-C4              C5            C6              C7");
printf("\n  %.3f             %.3f          %.3f            %.3f",xw1*K31,xw2*K32,xw3*K33,xw4*K34);
printf("\n Sum of Kxw = %d",xw1*K31+xw2*K32+xw3*K33+xw4*K34);
printf("\n Hence the still temperature Ts = 416 K");
printf("\n\n\n Calculation of feed condition");
printf("\n Component       xf             Tb = 377K              Tb = 376 K");
printf("\n                                     K          Kxf              K       Kxf ");
xf1 = 0.40;
xf2 = 0.23;
xf3 = 0.17;
xf4 = 0.20;
Kb1 = 1.80;            //equillibrium constants at 377 K for n-C4
Kb2 = 0.81;            //equillibrium constants at 377 K for C5
Kb3 = 0.39;            //equillibrium constants at 377 K for C6
Kb4 = 0.19;            //equillibrium constants at 377 K for C7
Kb11 = 1.78;           //equillibrium constants at 377 K for n-C4
Kb21 = 0.79;           //equillibrium constants at 377 K for C5
Kb31 = 0.38;           //equillibrium constants at 377 K for C6
```

```
Kb41 = 0.185;              //equillibrium constants at
   377 K for C7
printf("\n n-C4             %.2f       %.2f       %.3f
   %.2f    %.3f",xf1,Kb1,xf1*Kb1,Kb11,xf1*Kb11);
printf("\n C5               %.2f       %.2f       %.3f
   %.2f    %.3f",xf2,Kb2,xf2*Kb2,Kb21,xf2*Kb21);
printf("\n C6               %.2f       %.2f       %.3f
   %.2f    %.3f",xf3,Kb3,xf3*Kb3,Kb31,xf3*Kb31);
printf("\n C7               %.2f       %.2f       %.3f
   %.2f    %.3f",xf4,Kb4,xf4*Kb4,Kb41,xf4*Kb41);
printf("\n                                        %.3f
             %.3f",xf1*Kb1+xf2*Kb2+xf3*Kb3+xf4*Kb4
   ,xf1*Kb11+xf2*Kb21+xf3*Kb31+xf4*Kb41);

//Calculation of pinch temperatures
printf("\n\n\n The upper pinch temperature,Tn = %d K
   ",343+0.33*(416-343));
printf("\n The lower pinch temperature,Tm = %d K"
   ,343+0.67*(416-343));

//Calculation of approximate minimum reflux ratio.
printf("\n\n\n");
printf("\n Component        Tn = 367 K        Tm = 391 K
        xfh    axfh");
printf("\n                            a                  a
                    ");
printf("\n n-C4                     %.2f                %.2f
                  ",2.38,2.00);
printf("\n C5                       %.2f                %.2f
                  ",1.00,1.00);
printf("\n C6                       %.3f                %.3f
        %.2f      %.3f",0.455,0.464,0.17,0.077);
printf("\n C7                       %.3f                %.3f
        %.2f      %.3f",0.220,0.254,0.20,0.044);
printf("\n

   %.3f",0.077+0.044);
rf = xf1/xf2;
```

```
printf("\n rf = %.3f",rf);
xn4 = rf/[(1+rf)*(1+0.121)];
printf("\n xn4 = %.3f",xn4);
xn5 = xn4/rf;
printf("\n xn5 = %.3f",xn5);
Rm = [1/(2.38-1)]*(0.975/0.563)-2.38*(0.025/0.325);
printf("\n Rm = %.2f",Rm);

//The streams in the column
D = 40;
Ln = D*Rm;
Vn = Ln+D;
F = 100;
Lm = Ln + F;
W = 60;
Vm = Lm - W;
Ratio = Lm/W;
printf("\n Ln = %.1f kmol",44.8);
printf("\n Vn = %.1f kmol",84.8);
printf("\n Lm = %.1f kmol",144.8);
printf("\n Vn = %.1f kmol",84.4);
printf("\n Lm/W = %.2f",Ratio);
//Check on minimum reflux ratio
//xn = xd/(a-1)Rm
xn = xd1/[(2.38-1)*Rm];
printf("\n For n-C4......xn = %.3f",xn);
xn1 = 1-xn;
printf("\n For n-C5...xn = %.3f",xn1);
printf("\n Temperature check for upper pinch gives
    sum of K*xn = ");
sumKxn = 1.62*xn +0.68*xn1;
printf("%.3f",sumKxn);
```

Scilab code Exa 11.16 Minimum reflux ratio

```
//Example 11.16

clear;
clc;

printf("\tExample 11.16\n");

//Mole fractions
xf=[0.40 0.35 0.25];
xd=[0.534 0.453 0.013];
xw=[0 0.04 0.96];

//Amount of feed, product, bottom in kmol
F=[40 35 25];
D=[40 34 1];
W=[0 1 24];

//roots of equation
theta=[1.15 1.17];

//relative volatility
alpha=[2.7 2.22 1];

//Underwoods 1st equation for q=1
sums=[0 0];
for i=1:2
    for j=1:3
        sums(i)=sums(i)+(alpha(j)*xf(j)/(alpha(j)-
           theta(i)));
    end
end
printf("\nFrom Underwoods 1st eq\n");
printf("The value of 1-q at theta = 1.15 and 1.17
   are");
disp(sums);

//Underwoods 2nd equation for minimum reflux ratio
sum2=0;
```

```
for l=1:3
    sum2=sum2+(alpha(l)*xd(l)/(alpha(l)-theta(2)));
end

Rm=sum2-1;
printf("\nMinimum Reflux ratio is %.3f\n",Rm);

//End
```

Scilab code Exa 11.17 Number of theoretical plates required

```
//Example 11.17 Fenske's Equation

clear;
clc;

printf("\tExample 11.17\n");

//From previous question data
xA_d=0.453;
xB_d=0.013;

xA_s=0.04;
xB_s=0.96;

alpha_av=2.22;

//By Fenske Equation for no. of plates

n=((log(xA_d*xB_s/(xA_s*xB_d)))/log(alpha_av))-1;

printf("\nMinimum no. of plates are %f or %d\n",n,n)
    ;

//End
```

Scilab code Exa 11.18 Xhange in n with R

```
//Example 11.18

clear;
clc;

printf("\tExample 11.18\n");

R=[1 2 5 10];
Rm=0.83;

nm=8.5-1;

//X are points on X-axis of graph
for i=1:4
    X(i)=(R(i)-Rm)/(R(i)+1);
end

//Values at Y-axis for corresponding values of X-axis in graph are given by
Y=[0.55 0.32 0.15 0.08];

//where Y=(n+1)-(nm+1)/(n+2)


for i=1:4
    n=poly([0],'x');
    N(i)=roots(((n+1)-(nm+1))-(Y(i)*(n+2)));
end

printf("\nThe values of R and n are\n");
for i=1:4
    printf("\n\t%d \t %.2f",R(i),N(i));
```

```
end

printf("\n\nThe change in R and n can be seen as
    above\n");

//End
```

Scilab code Exa 11.19 Optimum reflux ratio

```
clear all;
clc;
printf("\n Example 11.19");
//Data from fig. 11.42
a = [0 0.02 0.04 0.06 0.08 0.1 0.2 0.4 0.6 0.8 1.0];
b = [0.75 0.62 0.60 0.57 0.55 0.52 0.45 0.30 0.18
    0.09 0];
//a = (R-Rm)/(R+1)
//b = [(n+1)-(nm+1)]/(n+2)
R = [0.92 1.08 1.25 1.75 2.5 3.5 5.0 7.0 9.0];
n = [28.6 22.8 16.9 13.5 11.7 10.5 9.8 9.2 8.95];
plot(n,R);
xtitle("Plot of R vs n","n","R");
printf("\n Derivative calculated from the graph");
d = [110.0 34.9 9.8 3.8 1.7 0.6 0.4 0.2 0.05];
i=1;
while i <=9
   s = R(i)+1 - (n(i)+7.72)/d(i);
   if s <=0.0001 then
       Ropt = R(i);
       printf("\n Ropt = %.2f",Ropt);
       break;
   end
   i=i+1;
end
printf("\n R is approximately %.1f percent of the
```

```
   minimum reflux condition",1.25/0.866666666*100);
```

Scilab code Exa 11.20 Plate efficiency for the given data

```
//Example 11.20

clear;
clc;

printf("\tExample 11.20\n");

//Mole fraction
xf=[0.2 0.3 0.2 0.3];

//Viscosity at mean tower temp. in mNs/m^2
uL=[0.048 0.112 0.145 0.188];

//Viscosity of water in mNs/m^2
uw=1;

sums=0;
for i=1:4
    sums=sums+(xf(i)*uL(i)/uw);
end

//Efficiency by DRICKAMER and BRADFORD
E=0.17-(0.616*log10(sums));

printf("\nEfficiency is %.2f",E);

//End
```

Chapter 12

Absorption of gases

Scilab code Exa 12.1 Overall liquid film coefficient

```
clear;
clc;
printf("\n Example 12.1");
//Overall liquid transfer coefficient KLa = 0.003
   kmol/s.m^3(kmol/m^3)

//(1/KLa)=(1/kLa)+(1/HkGa)
// let (KLa)=x
x = 0.003;
overall = 1/x;

//For the absorption of a moderately soluble gas it
   is reasonable to assume that the liquid and gas
   phase resistances are of the same order
   ofmagnitude, assuming them to be equal.
//(1/KLa)=(1/kLa)+(1/HkGa)
//let 1/kLa = 1/HkGa = y
y = (1/(2*x));
z = (1/y);            //z is in kmol/s m^3(kmol/m
   ^3)
printf("\n For S02:");
```

```
printf("\n    kGa = %f kmol/s m^3(kN/m^2)",z/50);
printf("\n For NH3:");
d_SO2 = 0.103;             //diffusivity at 273K for SO2 in cm^2/sec
d_NH3 = 0.170;             //diffusivity at 273K for NH3 in cm^2/sec
printf("\n    kGa = %f kmol/s m^3(kN/m^2)",(z/50)*(d_NH3/d_SO2)^0.56);
printf("\n For a very soluble gas such as NH3, kGa = KGa.");
printf("\n   For NH3 the liquid-film resistance will be small, and:");
printf("\n          kGa =KGa = %fkmol/s m^3(kN/m^2)",(z/50)*(d_NH3/d_SO2)^0.56);
```

Scilab code Exa 12.2 Mass transfer coefficient

```
clear;
clc;
printf("\n Example 12.2");
G = 2;                               //air flow rate in kg/m^2.sec
Re = 5160;                           //Re stands for Reynolds number
f = 0.02;                            //friction factor = R/pu^2
d_SO2 = 0.116*10^(-4);               //diffusion coefficient in m^2/sec
v = 1.8*10^(-5);                           //viscosity in mNs/m^2
p = 1.154;                           //density is in kg/m^3

//(hd/u)(Pm/P)(u/p/d_SO2)^0.56 = BRe^(-0.17)=jd

```

```
function[x]=a1()
    x = (v/(p*d_SO2))^(0.56);
    funcprot(0);
endfunction

//jd = f =R/pu^2
function[y]=a2()
    y = f/a1();
    funcprot(0);
endfunction
//G = pu
 u = (G/p);                          //u is in m/sec

function[x1]= a3()
    x1 = a2()*u;
    funcprot(0);
endfunction

function[d]=d1()
    d = Re*v/(G);
    funcprot(0);
endfunction

printf("\n d = %f mm",d1());
R = 8314;                //R is in m^3(N/m^2)/K kmol
T = 298;                 //T is in Kelvins
function[kG]=kG1()
    kG = a3()/(R*T);
    funcprot(0);
endfunction

printf("\n kG = %.2f*10^(-8) kmol/m^2sec(N/m^2)",kG1
   ()*10^(8));
printf("\n kG = %.2f*10^(-4)kg SO2/m^2sec(kN/m^2)",
   kG1()*10^(7)*64);
```

Scilab code Exa 12.3 Overall transfer units required

```
clear;
clc;
printf("\n Example 12.3");
//as the system are dilute mole fractions are approximately equal to mole ratios.
//At the bottom of the tower
y1 =0.015;                //mole fraction
G = 1;                    //Gas flow rate is in kg/m^2 sec
//At the top of the tower
y2 = 0.00015;             //mole fraction
x2 = 0;
L = 1.6;                 //liquid flow rate is in kg/m^2.sec
Lm = 1.6/18;             //liquid flow rate is in kmol/m^2.sec
Gm = 1.0/29;             //gas flow rate is in kmol/m^2.sec
x1 = poly([0],'x1');
x11 = roots(Gm*(y1-y2)-Lm*(x1));
printf("\n    x1 = %f",x11);
function[ye1]=henry_law(x)
    ye1 = 1.75*x;
    funcprot(0);
endfunction
bottom_driving_force = y1 - henry_law(x11);
 function[lm]=log_mean()
     lm = (bottom_driving_force-y2)/log(bottom_driving_force/y2);
     funcprot(0);
 endfunction
NoG = (y1-y2)/log_mean();
```

```
NoL = NoG*1.75*(Gm/Lm);
printf("\n      NoL =%.2f",NoL);
```

Scilab code Exa 12.4 Height of transfer units and number of transfer units

```
clear;
clc;
printf("\n Example 12.4");
Gm = 0.015;
KGa = 0.04;
top = 0.0003;
Y1 = 0.03;
Ye = 0.026;
bottom = (Y1-Ye);
log_mean_driving_force = (bottom-top)/log(bottom/top
   );
Z = poly([0],'Z');
Z1 = roots(Gm*(Y1-top)-KGa*log_mean_driving_force*Z)
   ;
printf("\n      Z =%.2fm",Z1);
HoG = Gm/KGa;
printf("\n  Height of transfer unit HoG = %.3fm",HoG
   );
printf("\n Number of transfer units NoG = %f",ceil
   (7.79/0.375));
```

Scilab code Exa 12.5 Height of the tower

```
clear;
clc;
printf("\n Example 12.5");
y1 = 0.10;
Y1 = 0.10/(1-0.10);
```

```
y2 = 0.001;
Y2 = y2;
mass_flowrate_gas = 0.95;        //mass flow rate in kg/m^2.sec
mass_percent_air = (0.9*29/(0.1*17+0.9*29))*100;
mass_flowrate_air = (mass_percent_air*mass_flowrate_gas);//in kg/m^2.sec
Gm = (mass_flowrate_air/29);
Lm = 0.65/18;           //Lm is in kmol/m^2.sec
//A mass balance between a plane in the tower where the compositions are X and Y and the top of the tower gives:

//Y = 1.173X+0.001
X = 0:0.001:0.159;
Y = 1.173*X+0.001;
plot(X,Y);

x=[0.021 0.031 0.042 0.053 0.079 0.106 0.159];

PG = [12 18.2 24.9 31.7 50.0 69.6 114.0];
i=1;
while i<8
    Y1(i) = PG(i)/(760-PG(i));
    i=i+1;
end
plot(x,Y1);
//xlabel("Area under the curve is 3.82m","kmolNH3/kmolH2O","kmolNH3/kmol air");
xtitle("Operating and equilibrium lines","kmol NH3/kmol H2O","kmol NH3/kmol air");
//If the equilibrium line is assumed to be straight, then:
//Gm(Y2      Y1) = KGaZdeltaPlm

//Top driving force
deltaY2 = 0.022;
//Bottom driving force
```

```
deltaY1 = 0.001;
deltaYlm = 0.0068;
Z = (0.0307*0.11)/(0.001*0.688);
printf("The height of the tower is %.2f meters",Z);
```

Scilab code Exa 12.6 specific steam consumption

```
clear;
clc;
printf("\n Example 12.6");
printf("\n Number of theoretical plates = %d"
    ,(30*0.3));

//At the bottom of the tower:
//Flowrate of steam = Gm (kmol/m^22s)
//Mole ratio of pentane in steam = Y1, and
//Mole ratio of pentane in oil = X1
X1 = 0.001;

//At the top of the tower:
//exit steam composition = Y2
//inlet oil composition = X2
X2 = 0.06;
//flowrate of oil = Lm (kmol/m^2.sec)

//The minimum steam consumption occurs when the exit
     steam stream is in equilibrium with the inlet
    oil, that is when:
//The equilibrium relation for the system may be
    taken as Ye = 3.0X, where Ye and X are expressed
    in mole ratios of pentane in the gas and liquid
    phases respectively.
Ye2 = X2*3;
//Lmin(X2      X1) = Gmin(Y2      Y1)
//If Y1 = 0, that is the inlet steam is pentane-free
```

```
    , then:
ratio_min = (X2 - X1)/Ye2;


//The operating line may be fixed by trial and error
    as it passes through the point (0.001, 0), and 9
    theoretical plates are required for the
    separation. Thus it is a matter of selecting the
    operating line which, with 9 steps, will give X2
    = 0.001 when X1 = 0.06. This is tedious but
    possible, and the problem may be better solved
    analytically since the equilibrium line is
    straight.

//let x = 1/A
//for a stripping operation
LHS = (X2-X1)/(X2);
printf("\n LHS = %f",LHS);
x = poly([0],'x');
x1 = roots((x^10-x)-LHS*(x^(10)-1));
printf("\n 1/A = %.2f",x1(9));
//A = Lm/mGm
printf("\n Gm/Lm = %.3f",x1(9)/3);
printf("\n Actual/minimum Gm/Lm = %.2f",0.457/0.328)
    ;

//If (actual Gm/Lm)/(min Gm/Lm) = 2,
printf("\n Actual Gm/Lm = %.3f",.328*2);
x2 = 3*(0.656);
printf("\n 1/A = mGm/Lm = %.3f",3*0.656);

//y = (1.968)^(N+1)
y = poly([0],'y');
y1 = roots(0.983*(y-1)-(y-1.968));
N = log(y1)/log(1.968)-1;
printf("\n The actual number of plates are : %.1f",
    ceil(N/0.3));
```

Chapter 13

Liquid liquid extraction

Scilab code Exa 13.1 Composition of final raffinate

```
clear all;
clc;
printf("\n Example 13.1");
printf("\n    (a) Countercurrent operation");
//(a) Countercurrent operation
S = 1.6*10^(-4);                    //Solvent flow rate in m^3/sec
mass_flowrate = (S*800);            //mass flow rate is in kg/sec

//Considering the solution, 400cm3/s = 4   10   4 m^3/sec containing, say, a m^3/sec A and (5   10   4   a) m3/sec B.
//Thus mass flow rate of A = 1200a kg/sec
//and mass flow rate of B = (4*10^(-4)-a)*1000 = (0.4-1000a)kg/sec
//a total of:                        (0.4+200a) kg/sec
C = poly([0],'C');
C1 = roots(0.1*(0.4+200*C)-1200*C);
printf("\n Concentration of the solution is %.2f*10^(-5) m^3/sec",C1*10^5);
```

```
printf("\n mass flow rate of A = %.3f kg/sec",1200*C1);
printf("\n mass flow rate of B =%.3f kg/sec",0.4+200*C1);
printf("\n Ratio of A/B in the feed , Xf = %.3f kg/kg",0.041/0.366);

X = [0.05 0.10 0.15];
Y = [0.069 0.159 0.258];
plot(X,Y);
xtitle("Equilibrium curve","kg A/kg B","kg A/kg S");
//From The curve:
slope = 0.366/0.128;
printf("\n Slope of the equilibrium line is %.2f",slope);

//Since pure solvent is added, Yn+1 = Y4 = 0 and a line of slope 2.86 is drawn in such that stepping off from Xf = 0.112 kg/kg to Y4 = 0 gives exactly three stages. When Y4 = 0, Xn = X3 = 0.057 kg/kg
printf("\n The composition of final raffinate is 0.057kg A/kg B");

printf("\n\n\n      (b)Multiple contact");
printf("\n Stage 1");
printf("\n In this case %.4f kg/sec",0.128/3);
//and from the equilibrium curve, the extract contains 0.18 A/kg S and (0.18   0.0427) = 0.0077 kg/s A.
printf("\n Thus raffinate from stage 1 contains %.4f kg/sec Aand %.3f kg/sec B",(0.041-0.0077),0.366);
X1 = 0.0333/0.366;
printf("\n X1 = %.3f kg/kg",0.0333/0.366);

printf("\n Stage 2");
//the extract contains 0.14 kg A/kg S
```

```
printf("\n The extract contains %.4f kg/sec A",0.14*0.0427);
//Thus: the raffinate from stage 2 contains (0.0333 0.0060) = 0.0273 kg/s A and 0.366 kg/s B
X2 = (0.0273/0.366);
printf("\n X2 = %.3f kg/kg",X2);

printf("\n Stage 3");
//the extract contains 0.114 kg A/kg S
printf("\n The extract contains %.4f kg/sec A/kg S",0.114*0.0427);
printf("\n Thus the raffinate contains %.4f kg/sec A and %.3f kg/sec B",(0.0273-0.0049),0.366);
printf("\n The composition of final raffinate = %3f kg A/kg B",0.0224/0.366);
```

Scilab code Exa 13.3 Overall transfer coefficient

```
clear all;
clc;
printf("\n Example 13.3");
//From the equilibrium relationship
CB1 = (0.0247*0.685);
printf("\n CB1* = %.4f kmol/m^3",CB1);
CB2 = (0.0247*0.690);
printf("\n CB2* = %.4f kmol/m^3",CB2);

//Thus the driving force at the bottom:
deltaC1 = (0.0169-0.0040);
printf("\n deltaC1 =%.4f kmol/m^3",deltaC1);
//Driving force at the top
deltaC2 = (0.0170-0.0115);
printf("\n deltaC2 = %.4f kmol/m^3",deltaC2);
function[x]= log_mean_driving_force()
    x = (deltaC1 - deltaC2)/log((deltaC1)/deltaC2);
```

```
    funcprot(0);
endfunction
printf("\n log mean driving force is given by
   deltaClm = %.4f kmol/m^3",log_mean_driving_force
   ());
KBa = (4.125*10^(-8))/(log_mean_driving_force()
   *0.0063);
printf("\n KBa = %.1f*10^(-4) kmol/sec m^3(kmol/m^3)
   ",KBa*10^4);
HoB = (1.27*10^(-3))/KBa;
printf("\n HoB = %.2f meters",HoB);
```

Scilab code Exa 13.4 Surface mean droplet size

```
clear all;
clc;
printf("\n Example 13.4");
diameter = [2 3 4 5 6];
number = [30 120 200 80 20];

function[x] = Sum_d1cube()
    sum = 0;
    i = 1;
    while (i <= 5)
        sum = sum + number(i)*(diameter(i))^3;
        i = i+1;
    end
    x = sum;
    funcprot(0);
endfunction

function[y]=sum_d1square()
    sum1 = 0;
    j=1;
    while(j<=5)
```

```
        sum1 = sum1 + number(j)*(diameter(j))^2;
        j= j+1;
    end
    y = sum1;
    funcprot(0);
endfunction

function[z]= ds()
    z = Sum_d1cube()/sum_d1square();
    funcprot(0);
endfunction
printf("\n Mean droplet size = %.2f mm",ds());
```

Scilab code Exa 13.5 Number of overall transfer units in raffinate phase

```
clear all;
clc;
printf("\n Example 13.5");
CSA = (%pi/4)*(0.075)^2;                    //cross
    sectional area is in m^2
V = (0.0044*3);                             //volume of
    packing is in m^3
C = 0.01;                                   //concentration
     is in kg/kg
printf("\n mass of acid transferred to the ether %.4
    f kg/m^2.sec or %f kg/sec",0.05*(0.01-0)
    ,0.005*0.0044);
printf("\n Acid in the aqueous feed = %.2f kg/m^3.
    sec",0.25*0.04);
printf("\n Acid in the raffinate = %.3f kg/m^2.sec"
    ,0.01-0.005);
printf("\n Concentration of acid in the raffinate =
    %.2f kg/kg",0.005/0.25);
printf("\n At the top of the column");
CR2 = 0.040;                     //Concentration is in kg/
```

```
kg
CR22 = 0.040*0.3;            //Concentration is in kg/kg
deltaC2 = (0.012-0.010);
printf("\n deltaC2 = %.3f kg/kg",deltaC2);
printf("\n\n At the bottom of the column");
CR1 = 0.20;                  //Concentration is in kg/kg
CR11 = 0.020*0.3             //Concentration is in kg/kg
deltaC1 = 0.006 -0;          //Concentration is in kg/kg
printf("\n deltaC1 = %.3f kg/kg",deltaC1);
deltaCRlm = (0.006-0.002)/log(0.006/0.002);
printf("\n Logarithmic driving force is : %.4f kg/kg",deltaCRlm);
KRa = 0.000022/(0.01333*deltaCRlm);
printf("\n KRa = %.3f kg/m^3.sec(kg/kg)",KRa);
printf("\n Height of an overall transfer unit = %.2f m",0.25/KRa);
printf("\n The number of overall transfer units = %.2f ",3/0.54);
```

Chapter 14

Evaporation

Scilab code Exa 14.1 Heat surface required

```
clear;
clc;
printf("\n Example 14.1");
//Assuming that the steam is dry and saturated at 205 kN/m2, then from the Steam Tables in the Appendix, the steam temperature = 394 K at which the total enthalpy = 2530 kJ/kg.

//At 13.5 kN/m2, water boils at 325 K and, in the absence of data on the boiling point elevation, this will be taken as the temperature of evaporation, assuming an aqueous solution. The total enthalpy of steam at 325 K is 2594 kJ/kg.

//Thus the feed, containing 10 per cent solids, has to be heated from 294 to 325 K at which temperature the evaporation takes place.

printf("\n mass of dry solids = %.1f kg/sec",(7*10/100));
x = poly([0],'x');
```

```
x1 = roots(0.7*100-50*(0.7+x));
printf("\n x = %.1f kg/sec",x1);
printf("\n Water to be evaporated = %.1f kg/sec",(7-0.7)-0.7);
printf("\n Summarising");
printf("\n Stream            Solids          Liquid           Total  ");
printf("\n                   (kg/s)          (kg/s)           (kg/s) ");
printf("\n Feed                 %.1f             %.1f              %.1f",x1,7-x1,x1+7-x1);
printf("\n Product              %.1f             %.1f             %.1f",x1,x1,x1+x1);
printf("\n Evaporation                           %.1f             %.1f",7-x1-x1,7-2*x1);
//Using a datum of 273K
q_entering = (7*3.76)*(294-273);
printf("\n Heat entering with the feed = %.1f kW",q_entering);
q_leaving = (1.4*3.14)*(325-273);
printf("\n Heat leaving with the product = %.1f kW",q_leaving);
printf("\n Heat leaving with the evaporated water = %d kW",5.6*2594);
printf("\n Heat transferred from the steam = %d kW",14526+228.6-552.7);
printf("\n The enthalpy of the condensed steam leaving at 352.7 K = %.1f kJ/kg ",4.18*(352.7-273));
printf("\n The heat transferred from 1 kg steam = %.1f kJ/kg",2530-333.2);
printf("\n Steam required = %.2f kg/s ",14202/2196.8);

//s the preheating of the solution and the sub-cooling of the condensate represent but a small proportion of the heat load, the temperature driving force may be taken as the difference
```

```
   between the temperatures of the condensing steam
   and the evaporating water, or:
printf("\n deltaT = %d deg K ",394-325);
printf("\n Heat transfer area ,A = %.1f m^2"
   ,14202/(3*69));
```

Scilab code Exa 14.2A Forward feed

```
clear;
clc;
printf("\n Example 14.2a");
//Temperature of dry saturated steam at 205 kN/m2 =
   394 K.
//At a pressure of 13 kN/m2 (0.13 bar), the boiling
   point of water is325 K, so that the
   totaltemperature difference &T = (394     325) =
   69deg K.



//First Approximation.

//Assuming that: U1*deltaT1 = U2*deltaT2 = U3*
   deltaT3
//then substituting the values of U1, U2 and U3 and
   &T = 69 deg K gives:
printf("\n deltaT1 = %f deg K",13);
printf("\n deltaT2 = %f deg K",20);
printf("\n deltaT2 = %f degK",36);

//Since the feed is cold, it will be necessary to
   have a greater value of T1 than given by this
   analysis. It will be assumed that T1 = 18 deg K,
   T2 = 17 deg K, T3 = 34 deg K.

//For steam to 1: T0 = 394 K and   0 = 2200 kJ/kg
```

```
//For steam to 2: T1 = 376 K and   1 = 2249 kJ/kg
//For steam to 3: T2 = 359 K and   2 = 2293 kJ/kg
//                T3 = 325 K and   3 = 2377 kJ/kg

//Assuming that the condensate leaves at the steam
   temperature, then heat balances across each
   effect may be made as follows:

//Effect 1 :
//2200 D0 = 4    4.18(376     294) + 2249 D1

//Effect 2:
//2249 D1 + (4     D1)4.18(376     359) = 2293 D2

//Effect 3:
//2293 D2 + (4     D1     D2)4.18(359     325) =
   2377 D3

printf("\n            Solids              liquor
                 Total  ");
printf("\n            (kg/s)              (kg/s)
                 (kg/s) ");
printf("\nFeed        %.1f                %.1f
                   %.1f  ",0.4,3.6,4.0);
printf("\nProduct     %.1f                %.1f
                   %.1f  ",0.4,0.4,0.8);
printf("\nEvaporation                     %.1f
                   %.1f  ",3.6-0.4,4.0-0.8);

//D1 + D2 + D3 = 3.2 kg/s
D = [2177.94 -2293 0;-4.18*(359-325)
   2293-4.18*(359-325) -2377;1 1 1];
C = [-4*4.18*(376-359);-4*4.18*(359-325);3.2];
D1 = inv(D)*C;
printf("\n ans = %.3f kg/s",D1);

D0 = (4*4.18*(376-294)+2249*D1(1))/2200;
A1 = D0*2200/(3.1*18);
```

```
printf("\n A1 = %.1f m^2",A1);
A2 = D1(1)*2249/(2*17);
printf("\n A2 = %.1f m^2",A2);
A3 = D1(2)*2293/(1.1*34);
printf("\n A3 = %.1f m^2",A3);
printf("\n These three calculated areas are
    approximately equal, so that the temperature
    differences assumed may be taken as nearly
    correct");

//let, deltaT1 = x
//deltaT2 = y
//deltaT3 = z
X = [3.1 -2.0 0;0 2.0 -1.1;1 1 1];
A = [0;0;69];
T = inv(X)*A;
printf("\n deltaT1 = %d deg K",T(1));
printf("\n deltaT2 = %d deg K",T(2));
printf("\n deltaT3 = %d deg K",T(3));

//ghting the temperature differences to allow for
    the fact that the feed enters at ambient
    temperature gives:
//deltaT1 = 18 deg Kdelta T2 = 18 degK delta T3 = 33
     deg K
Steam_temp = 394;
T1 = Steam_temp-18;
printf("\n T1 = %d K",T1);
T2 = T1 - 18;
printf("\n T2 = %d K",T2);
T3 = T2 - 33;
printf("\n T3 = %d K",T3);

//The total evaporation (D1 + D2 + D3) is obtained
    from a material balance:
printf("\n          Solids                liquor
              Total   ");
printf("\n          (kg/s)                (kg/s)
```

```
(kg/s) ");
printf("\nFeed        %.1f                    %.1f                %.1f   ",0.4,3.6,4.0);
printf("\nProduct     %.1f                    %.1f                %.1f   ",0.4,0.4,0.8);
printf("\nEvaporation                          %.1f                %.1f   ",3.6-0.4,4.0-0.8);

//Assuming, as an approximation, equal evaporation in each effect, or D1 = D2 = D3 = 1.07 kg/s, then  the latent heat of flash vaporisation in the second effect is given by:
q = 4.18*(4-1.07)*(376-358);
printf("\n latent heat of flash vaporisation in the second effect is = %.1f kW",q);
printf("\n latent heat of flash vaporisation in the third effect is:= %.1f kW ",4.18*(4-2*1.07)*(358-325));
printf("\n At 394 K, the latent heat = 2200 kJ/kg");
printf("\n At 325 K, the latent heat = 2377 kJ/kg");
printf("\n Mean value,     = 2289 kJ/kg");

printf("\n Values ofT1,T2T3 are now chosen by trial and error to give equal values of A in each effect, as follows");
printf("\n deltaT1     A1     deltaT2     A2        deltaT3       A3");
printf("\n (deg K)    (m^2)   (deg K)    (m^2)    (deg K)       m^2");
printf("\n    18        64.2      18          61.4       33        66.9 ");
printf("\n    19        60.5      17          65.0       33        66.9");
printf("\n    18       64.2     17.5        63.1       33.5     65.9");
printf("\n    18       64.2     17          65.0       34        64.9");

```

```
printf("\n Steam consumption = %.2f kg/s",3580/2289);
printf("\n Economy = %.1f kg/kg",3.2/1.56);
```

Scilab code Exa 14.2B Backward feed

```
clear;
clc;
printf("\n Example 14.2b");
deltaT1 = 20;               //temperature is in deg K
deltaT2 = 24;               //temperature is in deg K
deltaT3 = 25;               //temperature is in deg K
To = 394;                   //temperature is in deg K
T1 = 374;                   //temperature is in deg K
T2 = 350;                   //temperature is in deg K
T3 = 325;                   //temperature is in deg K
latent_heat0 = 2200;        //latent heat in kJ/kg
latent_heat1 = 2254;        //latent heat in kJ/kg
latent_heat2 = 2314;        //latent heat in kJ/kg
latent_heat3 = 2377;        //latent heat in kJ/kg

//Effect 3:
//            2314 D2 = 4   4.18(325   294) + 2377 D3

//Effect 2:
//            2254 D1 = (4   D3)4.18(350   325) + 2314 D2

//Effect 1:
//            2200 D0 = (4   D3   D2)4.18(374   350) + 2254 D1

//(D1 + D2 + D3) = 3.2 kg/s

```

```
D = [0 2314 -2377;2254 -2314 4.18*(350-325);1 1 1];
C = [4*4.18*(325-294);4*4.18*(350-325);3.2];
D1 = inv(D)*C;
printf("\n D1 = %.3f kg/s",D(1));
printf("\n D2 = %.3f kg/s",D(2));
printf("\n D3 = %.3f kg/s",D(3));
A1  = 1.387*latent_heat0/(2.5*20);
A2 = 1.261*latent_heat1/(2*24);
A3 = 1.086*latent_heat2/(1.6*25);
printf("\n A1 = %.1f m^2",A1);
printf("\n A2 = %.1f m^2",A2);
printf("\n A3 = %.1f m^2",A3);
```

Scilab code Exa 14.3 Efficiency of the compressor

```
clear;
clc;
printf('\n Example 14.3");
He = 2690;          //He is the enthalpy of entrained
    steam in kJ/kg
H4 = ((1*2780)+(1.6*2690))/2.60;

//Again assuming isentropic compression from 101.3
    to 135 kN/m2, then:
H3 = 2640;        //in kJ/kg (from the chart)
n = (1.0+1.6)*(2725-2640)/[1.0*(2780-2375)];
printf("\n    = %.2f ",n);
printf("\n This value is low, since in good design
    overall efficiencies approach 0.75  0.80.
    Obviously the higher the efficiency the greater
    the entrainment ratio or the higher the saving in
     live steam");
Pe = 101.3;         //pressure of entrained vapour in
    kN/m^2
discharge_P = 135;//discharge pressure in kN/m^2
```

```
printf("\n the required flow of live steam = 0.5 kg/kg entrained vapour.");
printf("\n In this case the ratio is (1.0/1.6) = 0.63 kg/kg");
```

Scilab code Exa 14.4 Quantity of additional stream required

```
clear;
clc;
printf("\n Example 14.4");
//Making a mass balance, there are two inlet streams -
//the additional steam, say Gx kg/s
//the sea water feed, say Gy kg/s.

//The two outlet streams are
//distilled water product, 0.125 kg/s
//the concentrated sea water, 0.5 Gy kg/s

//Thus: (Gx + Gy) = (0.125 + 0.5 Gy) or (Gx + 0.5 Gy) = 0.125

//At 650 kN/m2, the total enthalpy of the steam = 2761 kJ/kg.

//Thus the energy in this stream = 2761Gx kW
//sea water enters at 344 K
//enthalpy of feed = [Gy   4.18(344   273)] = 296.8Gy kW
printf("\n The enthalpy of the product is 439*0.125 = 54.9 kW");

//Making a balance
//(E + 2761Gx + 296.8Gy) = (211.3Gy + 54.9)
//(E + 2761Gx + 85.5Gy) = 54.9
```

```
//where E is the power supplied to the compressor
//(E + 2590Gx) = 33.5

//For a single-stage isentropic compression
//the work done in compressing a volume V1 of gas at pressure P1 to a volume V2 at pressure P2 is given by
P1 = 101.3;             //pressure is in kN/m^2
P2 = 120;               //pressure is in kN/m^2
dsteam_P1 = (18/22.4)*(273/374.1); //density of steam at 101.3 kN/m^2 and 374.1 K
//V1 = 0.5Gy/dsteam_P1 = 0.853 Gy m^3/sec
E = poly([0],'E');
E1 = roots(0.5*E-(P1*0.853)/(1.3-1)*[(P2/P1)^(0.3/1.3)-1]);
printf("\n E = %.1f kW/(kg/sec)",E1);

printf("\n As the compressor is 50 per cent efficient then E = %.1f kW/(kg/sec)",E1/0.5);
//E = 46*0.5*Gy = 23.0Gy kW
Gx = poly([0],'Gx');
Gx1 = roots(2761*Gx - 54.9 - 4.72*(33.5-2590*Gx));
printf("\n Gx = %.3f kg/sec",Gx1);
```

Scilab code Exa 14.5 Capacity and economy of the system

```
clear;
clc;
printf("\n Example 14.5");
//Properties of the ejector
e1 = 0.95;            //nozzle efficiency
e2 = 0.80;            //efficiency of momentum transfer
e3 = 0.90;            //efficiency of compression
P1 = 650;             //pressure of live stream
```

```
P2 = 101.3;          //pressure of entrained steam
H1 = 2970;//The enthalpy of the live steam at 650 kN/m2 and (435 + 100) in  kJ/kg
H2 = 2605;//the enthalpy after isentropic expansion from 650 to 101.3 kN/m2, using an enthalpy entropy chart,in kJ/kg
x2 = 0.97;          //dryness fraction

printf("\n The enthalpy of the steam after actual expansion to 101.3 kN/m^2 is %d kJ/kg",e1*(H1 - H2));
LatentHeat = 2258;       //latent heat at 101.3 kN/m^2
//dryness after expansion but before entrainment x21 is given by:
//(x21  x2)  = (1    e1)(H1    H2)
x21 = poly([0],'x21');
x22 = roots((x21-x2)*LatentHeat-(1-e1)*(H1-H2));
printf("\n dryness after expansion but before entrainment x21 is = %.3f",x22);

//If x23 is the dryness after expansion and entrainment, then:
//(x23  x2)  = (1    e3)(H1    H2)
x23 = poly([0],'x23');
x24 = roots((x23-x2)*LatentHeat-(1-e3)*(H1-H2));
printf("\n the dryness after expansion and entrainment is %.2f",x24);

P3 = 205;              //discharge pressure in kN/m^2
//Assuming that the steam at the discharge pressure P3 = 205 kN/m2 is also saturated
x3 = 1;
H3 = 2675;//enthalpy of the mixture at the start of compression in the diffuser section at 101.3 kN/m2

//Again assuming the entrained steam is also
```

```
    saturated, the enthalpy of the mixture after
    isentropic compression in thediffuser from 101.3
    to 205 kN/m2 is H4
H4 = 2810;        //in kJ/kg
m = {[(H1-H2)/(H4-H3)]*e1*e2*e3-1};   //m = m2/m1
printf("\n m2/m1 = %.2f kg vapour entrained/kg kive
    stream",m);
printf("\n Thus with a flow of 0.14 kg/s live steam,
     the vapour entrained at 101.3 kN/m2 is %.2f kg/
    sec",0.14*m);
printf('\n Total saturated steam = %.2f kg/sec at
    205 kN/m^2",0.14+0.14*m);
printf("\n The economy of the steam = %.2f
    ",0.214/0.14);
printf("\n The capacity in terms of throughput of
    solution Gf = %.3f kg/sec",0.214+0.025);
```

Scilab code Exa 14.6 Method to drive the compressor

```
clear;
clc;
printf("\n Example 14.6");
//(a) Diesel engine
printf("\n (a) Diesel engine");

printf("\n For 1 kg evaporation, ammonia circulated
    = 2.28 kg");
printf("\n Work done in compressing the ammonia = %d
     MJ/kg",150*2.28);
printf("\n For an output of 1 MJ, the engine
    consumes 0.4 kg fuel.");
printf("\n fuel consumption = %.3f kg/kg water
    evaporated",0.4*0.342);
printf("\n cost = %.5f euro/kg water evaporated "
    ,0.02*0.137);
```

```
printf("\n (b) Turbine");

printf("\n The work required is 0.342 MJ/kg evaporation");
//Therefore with an efficiency of 70 per cent:
printf("\n energy required from steam = %.3f MJ/kg ",0.342*100/70);
printf("\n Enthalpy of steam saturated at 700 kN/m2 = 2764 kJ/kg.");
printf("\n Enthalpy of steam saturated at 101.3 kN/m2 = 2676 kJ/kg");
printf("\n thus : energy from steam = %d kg/kg",2764-2676);
printf("\n steam required = %.2f kg/kg evaporation ",0.489/0.088);
printf("\n Cost = %.4f euro/kg water evaporated",0.01*5.56/10);
printf("\n Hence the Diesel engine would be used for driving the compressor");
```

Scilab code Exa 14.7 Optimum boiling time

```
clear;
clc;
printf("\n Example 14.7");
//(a) Maximum throughput

printf("\n (a) Maximum throughput");
t_bopt = (15*10^3)+(2/(7*10^(-5)))*(7*10^(-5)*0.2*15*10^3)^0.5;
printf("\n The boiling time to give maximum heat transfer and hence maximum throughput is %.1f ksecs ",t_bopt*10^(-3));

```

```
Qb = (2*40*40)*(7*10^(-5))*[((7*10^(-5)*2.81*10^4)+0.2)^0.5-0.2^0.5];
printf("\n The heat transferred during boiling is %.2f*10^7 kJ",Qb);
printf("\n the water vaporated = %.2f*10^4 kg",(4.67*10^(7)/2300)*10^(-4));
printf("\n Rate of evaporation during boiling = %.3f kg/secs",(2.03*10^4)/(2.81*10^4));
printf("\n Mean rate of evaporation during the cycle = %.3f kg/secs",2.03*10^(4)/[(2.8*10^4)+(15*10^3)]);
printf("\n cost = %.1f euro/cycle",((2.81*10^(4)*18)/1000)+600);


printf("\n (b) Minimum cost");
t_bopt1 = (600/0.018)+[2*(7*10^(-5)*0.2*600*0.018)^0.5]/(7*10^(-5)*0.018);
printf("\n The boiling time to give minimum cost is %.1f ksecs",t_bopt1*10^(-3));
Qb1 = [(2*40*40)/(7*10^(-5))]*[(7*10^(-5)*5.28*10^4 + 0.2)^0.5-0.2^0.5];
printf("\n The heat transferred during one boiling period is %.2f*10^7 kJ",Qb1*10^(-7))
printf("\n The water evaporated = %.2f*10^4 kg",Qb1/2300);
printf("\n Rate of evaporation = %.3f kg/secs",(3.03/5.28));
printf("\n Mean rate of evaporation during the cycle = %.2f kg/secs",(3.03*10^4)/[(5.28*10^4)+(15*10^3)]);
printf("\n cost of one cyce = %.1f euro",5.28*10^(4)*0.018+600);
```

Chapter 15

Crystallisation

Scilab code Exa 15.1 Supersaturation ratio

```
clear;
clc;
printf("\n Example 15.1");
printf("\n For concentrations in kg sucrose/kg water
    :");
c = 2.45;                    //concentration is in kg/kg
printf("\n c = %.2f kg/kg",c);
c1= 2.04;                    //concentration is in kg/kg
printf("\n c1 = %.2f kg/kg",c1);
S = c/c1;
printf("\n S = %.2f",S);

printf("\n\n For concentrations in kg sucrose/kg
    water:")
co = c/(c+1);                //concentration is in kg/kg
     solution
printf("\n co = %.3f kg/kg solution",co);
co1 = c1/(c1 + 1);           //concentration is in kg/kg
     solution
printf("\n co1 = %.3f kg/kg solution",co1);
S = co/co1;
```

```
printf("\n S = %.2f ",S);
```

Scilab code Exa 15.2 Increase in solubility

```
clear;
clc;
printf("\n Example 15.2");
printf("\n For barium sulphate");

function[x] = increase_barium(d)
    y = exp((2*233*0.13)/(2*8314*298*4500*d));
    x = y-1;
    funcprot(0);
endfunction
i=1;
d = [0.5 0.05 0.005];
while(i<=3)
    printf("\n Increase in solubility for barium
       sulphate fpr particle size %f um = %f per
       cent",d(i)*2,(increase_barium(d(i)*10^(-6))
       *100));
    i = i+1;
end

printf("\n For Sucrose");
function[a] = increase_sucrose(d1)
    b = exp((2*342*0.01)/(1*8314*298*1590*d1));
    a = b-1;
    funcprot(0);
endfunction
j=1;
while(j<=3)
    printf("\n Increase in solubility for barium
       sulphate fpr particle size %f um = %f per
       cent",d(j)*2,(increase_sucrose(d(j)*10^(-6))
```

```
        *100));
    j = j+1;
end
```

Scilab code Exa 15.3 Theoretical yield

```
clear;
clc;
printf("\n Example 15.3");
R = 322/142;
//The initial concentration
c1 = 1000/5000;                    //Concentration is in
   kg Na2SO4/kg water
printf("\n The initial concentration c1 = %.1f kg
   Na2SO4/kg water",c1);
//The solubility
c2 = 9/100;                        //solubility is in Kg
   Na2SO4/kg water
printf("\n The solubility c2 = %.2f Kg Na2SO4/kg
   water",c2);
printf("\n Initial mass of water,w1 = 5000 kg and
   the water lost by evaporation E = %f kg/kg"
   ,2/100);
printf("\n yield y = %d kg Na2SO4.10H2O",(5000*2.27)
   *[0.2-0.09*(1-0.02)]/[1-0.09*(2.27-1)]);
```

Scilab code Exa 15.4 Yield of Sodium acetate

```
clear;
clc;
printf("\n Example 15.4");
//Given data:
//Heat of crystallisation, q = 144 kJ/kg trihydrate
```

```
//Heat capacity of the solution, Cp = 3.5 kJ/kg deg K
//Latent heat of water at 1.33 kN/m2,    = 2.46 MJ/kg
//Boiling point of water at 1.33 kN/m2 = 290.7 K
//Solubility of sodium acetate at 290.7 K, c2 = 0.539 kg/kg water

//Equilibrium liquor temperature
T = 290.7+11.5;
printf("\n Equilibrium liquor temperature is%.1f K",T);

//Initial concentration
c1 = 40/(100-40);
printf("\n Initial concentration c1 = % .3f kg/kg water",c1);

//Final concentration
c2 = 0.539;
printf("\n Final concentration,c2 = %.3f kg/kg water",c2);

//Ratio of molecular masses
printf("\n Ratio of molecular masses,R = %.2f",136/82);

E = {144*1.66*(0.667-0.539)+3.5*(353-302.2)*(1+0.667)*[1-0.539*(1.66-1)]}/{2460*[1-0.539*(1.66-1)]-(144*1.66*0.539)};
printf("\n\n\n E = %.3f kg/kg water originally present",E);
printf("\n yeild y = %.3f kg/sec",(0.56*(100-40)/100)*1.66*[0.667-0.539*(1-0.153)]/[1-0.539*(1.66-1)]);
```

Scilab code Exa 15.5 Length of crystalliser

```
clear;
clc;
//The molecular mass of hydrate/molecular mass of anhydrate
 R = 380/164;
printf("\n The molecular mass of hydrate/molecular mass of anhydrate,R = %.2f",R);
//It will be assumed that the evaporation is negligible and that E = 0.
c1 = 0.23;
printf("\n The initial concentration, c1 = %.2f kg/kg solution or %.2f kg/kg water",c1,c1/(1-c1));
c2 = 15.5;
printf("\n The final concentration, c2 = %.1f kg/kg water or 0.155 kg/kg water",c2);
w1 = 0.77;
printf("\n In 1 kg of the initial feed solution, there is 0.23 kg salt and 0.77 kg water and hence w1 = %.2f kg",w1);
y = 2.32*0.77*[0.30-0.155*(1-0)]/[1-0.155*(2.32-1)];
printf("\n yeild y = %.2f kg",y);
printf("\n In order to produce 0.063 kg/s of crystals, the required feed is: %.3f kg/sec",1*0.063/0.33);
q = 0.193*3.2*(313-298);
printf("\n The heat required to cool the solution %.1f kW",q);
q1 = 0.063*146.5;
printf("\n Heat of crystallisation = %.1f kW",q1);
printf("\n total heat loss = %.1f kW",q+q1);
printf("\n Assuming counter current flow");
deltaT1 = (313-293);
```

```
printf("\n deltaT1 = %d deg K",deltaT1);
deltaT2 = 298 - 288;
printf("\n deltaT2 = %d deg K",deltaT2);
deltaTlm = (deltaT1-deltaT2)/log(deltaT1/deltaT2);
printf("\n The logarithmic mean temperature is %.1f deg K",deltaTlm)

A = (q+q1)/(0.14*14.4);
printf("\n The heat transfer area required , A= Q/U
deltaTm = %.1f m^2",A);
printf("\n Length of heat exchanger required = %.1f",A)

printf("\n\n\n\n In practice 3 lengths , each of 3 m length would be specified.");
```

Scilab code Exa 15.6 Crystal production rate

```
clear;
clc;
printf("\n Example 15.6");
down_time = (4/0.00017);
printf("\n Draw down time = %d secs",down_time);
printf("\n\n\n                (a)");
printf("\n from a mass balance the total mass of solids is :")
cs = (6*0.6*2660*10^13)*(10^(-8)*23530)^4;
printf("\n cs = %d kg/m^3",cs);
printf("\n\n\n                (b)");
printf("\n The production rate = %.3f kg/sec",cs*0.00017);

printf("\n The crystal population decreases exponentially with size ")
```

```
x = exp((-100*10^(-6))/(10^(-8)*23530));
printf("%d percent",x*100);
printf("\n Thus 34 percent have been discharged by the time they reach 100 um");
printf("\n\n\n                    (c)");
//If (100      90) = 10 per cent of the nuclei remain and grow to >100   m , then
t = poly([0],'t');
t1 = roots(t - 10^(4)/log(1/0.10));
printf("\n tr = %d secs",t1);
Q = poly([0],'Q');
Q1 = roots(Q + (4/4343-0.00017));
printf("\n Qf = %f m^3/sec",-Q1);
```

Scilab code Exa 15.7 Vapour pressure

```
clear;
clc;
printf("\n Example 15.7");
l = poly([0],'l');
l1 = roots(log(780/220)-[l*(463-433)]/[8314*463*433]);
printf("\n   v  = %d kj/kmol",l1);
P = poly([0],'P');
P1 = roots(P-220/exp(70340*(433-393)/(8.314*433*393)));
printf("\n   P = %d kN/m^2",P1);
```

Scilab code Exa 15.8 Mass sublimation rates

```
clear;
clc;
printf("\n Example 15.8");
```

```
printf("\n Vaporisation stage :");
Pt = 101.5;
Ps = 1.44;
Sv = 0.56*(138/29)*(1.44/(101.5-1.44));
printf("\n Sv = %.3f kg/sec",Sv);

printf("\n\n COndensation stage: ")
Ptc = 100;
Psc = 0.0023;
S = 0.56*(138/29)*(0.0023/(100-0.0023));
printf("\n S = %f kg/sec",S);
```

Chapter 16

Drying

Scilab code Exa 16.1 Time taken to dry the solids

```
clear;
clc;
printf("\n Example 16.1");
//For the first drying operation
w1 = 0.25;                          //in kg/kg
w = 0.10;                           //in kg/kg
wc = 0.15;                          //in kg/kg
we = 0.05;                          //in kg/kg
f1 = w1-we;                         //in kg/kg
fc = wc - we;                       //in kg/kg
f = w - we;                         //in kg/kg

//x = mA
function[x]=TotalDryingTime(t)
    x = (1/t)*[(f1-fc)/fc + log(fc/f)];
    funcprot(0);
endfunction
printf("\n     mA = %.3f kg/sec",TotalDryingTime(15)
   );

//For the second drying operation
```

```
w12 = 0.30;                    //in kg/kg
w2 = 0.08;                     //in kg/kg
wc2 = 0.15;                    //in kg/kg
we2 = 0.05;                    //in kg/kg
f12 = w12 - we2;               //in kg/kg
fc2 = wc2 - we2;               //in kg/kg
f2 = w2 - we2;                 //in kg/kg

t1=(1/TotalDryingTime(15))*[(0.25-0.10)/0.10 + log
    (0.10/0.03)];
printf("\n The total drying time is then %.1f ksec
    or %.2f hr ",t1,t1*1000/3600);
```

Scilab code Exa 16.2 Time taken to dry the solids

```
clear;
clc;
printf("\n Example 16.2");
E = [1 0.64 0.49 0.38 0.295 0.22 0.14];
J = [0 0.1 0.2 0.3 0.5 0.6 0.7];
plot(J,E,rect=[0,1,0,1]);
xtitle("Plot for drying data","J = kt/L^2","E");

//For the 10 mm strips
mi = (0.28 - 0.07);          //Initial free moisture
    content in kg/kg
mf = (0.13-0.07);            //Final free moisture
    content in kg/kg
//at
t = 25;                      //time is in ksecs
E = (0.06/0.21);
//at E = 0.286 ,J = 0.52 from the plot of given data
     and J = kt/L^2
k = poly([0],'k');
k1 = roots(0.52 - (k*t)/(10/2)^2);
```

```
printf("\n k = %f",k1);

//for the 60 mm planks
m1i = (0.22 - 0.07);        //Initial free moisture content in kg/kg
m1f = (0.13 - 0.07);        //Final free moisture content in kg/kg
E = (m1f/m1i);
//at E = 0.20 from yhe plot of the given data J = 0.63 and J = kt/L^2
t1 = 0.63*(60/2)^(2)/k1;
printf("\n t1 = %d ksecs or %.1f days",t1,(t1*1000/(3600*24)));
```

Scilab code Exa 16.3 Mass flow rate of dry air

```
clear;
clc;
printf("\n example 16.3");
//(a) Air
//G kg/s dry air enter with 0.006G kg/s water vapour  and hence the heat content of this stream=
//[(1.00G) + (0.006G    2.01)](385     273) = 113.35 G kW

//(b) Wet solid
//0.125 kg/s enter containing 0.40 kg water/kg wet solid , assuming the moisture is expressed on a wet basis.
flowWater = (0.125*0.40);                //in kg/sec
flowDrySolid = (0.125-0.050);            //in kg/sec
//Hence heat content of this stream
q = [(0.050*4.18)+(0.075*0.88)]*(295-273);
printf("\n The heat content of this stream = %.2f kW",q);
```

```
//Heat out
//(a) Air
//Heat in exit air = [(1.00 G) + (0.006 G   2.01)](310   273) = 37.45G kW.
fd = 0.075;              //mass flow rate of dry solids in kg/sec
w = 0.05*0.075/(1+0.05);  //water in the dried solids leaving in kg/secs
we = (0.050 - w);         //The water evaporated into gas stream in kg/secs

//Assuming evaporation takes place at 295 K,then:
qout = 0.0464*[2.01*(310-295)+2449+4.18*(295-273)];
printf("\n Heat in the water vapour = %.1f kW",qout);

//the total heat in this stream = (119.30 + 37.45G) kW.
//(b) Dried solids


   //The dried solids contain 0.0036 kg/s water and hence heat conten   t of this stream is:
   q1 = [(0.075*0.88)+(0.0036*4.18)/(305-273)];
   printf("\n The dried solids contain 0.0036 kg/s water and hence heat content of this stream is: %.2f kW",q1);


//(c) Losses
//These amount to 20 kJ/kg dry air or 20m kW.
//Heat Balance
G = poly([0],'G');
G1 = roots(113.35*G + 6.05 - 119.30 - 37.45*G - 2.59 - 20*G);
printf("\n G = %.2f kg/secs",G1);
printf("\n Water in the outlet stream %.4f kg/secs"
```

```
    ,0.006*2.07+0.0464);
printf("\n The humidity H = %.4f kg/kg dry air"
    ,0.0588/2.07);
```

Scilab code Exa 16.4 Approximate drying time

```
clear;
clc;
printf("\n Example 16.4");
//In 100 kg of feed

    //mass of water =
    mw = 100*30/100;                    //mass of water
        in kg
    //mass of dry solids =
    md = 100-30;                        //mass of dry
        solids

//and:
//For b kg water in the dried solids: 100b/(b + 70)
    = 15.5
b = poly([0],'b');
b1 = roots(100*b - 15.5*(b+70));
printf("\n water in the product ,b = %.1f kg",b1);

    //Initial water content
    w1 = 30/70;             //Intial moisture content
        in kg/kg dry solids
    //Final moisture content
    w2 = (12.8/70);         //Final moisture content
        in kg/kg dry solids
    //water to be removed
    w3 = (30-12.8);         //water to be removed in
        kg

```

```
//Surface of drying
S = (0.03*70);              //Surface for drying in m
    ^2
rate = (0.0007*2.1);        //Rate of drying in kg/sec

//As the final moisture content is above the
    critical value, all the drying is at this
    constant rate and the time of drying is:
t = 17.2/0.00147;
printf("\n As the final moisture content is above
    the critical value, all the drying is at this
    constant rate and the time of drying is: %d ksecs
     or %.2f hr",t,t/3600);
```

Scilab code Exa 16.5 Proposed diameter and length

```
clear;
clc;
printf("\n Example 16.5");
//Inlet air temperature
Tair = 400;               //Inlet air temperature in
    kelvins
H = 0.01;                 //Humidity is in kg/kg dry
    air
//*therefore wet bulb temperature =
Twetbulb = 312;           //Inlet wet-bulb temperature
NTU = 1.5;                 //Number of transfer units

//Then for adiabatic drying the outlet air
    temperature,To is given by
To = poly([0],'To');
To1 = roots(exp(1.5)*(To-312)-(400-312));
printf("\n For adiabatic drying the outlet air
    temperature = %.1f K",To1);

```

```
//Solids outlet temperature will be taken to be maximum allowable ,325K
//From the steam tables in the Appendix , the latent heat of vaporisation of water at 312 K is2410 kJ/kg. Again from steam tables , the specific heat capacity of water vapour = 1.88 kJ/kg K and that of the solids will be taken as 2.18 kJ/kg K.

//For a mass flow of solids of 0.35 kg/s and inlet and outlet moisture contents of 0.15 and 0.005 kg/kg dry solids respectively , the mass of water evaporated = 0.35(0.15    0.005) = 0.0508 kg/s.

//For unit mass of solids ,the heat duty includes:
        //Heat to the solids
        qsolids = 2.18*(325-300);        //heat to solids in kJ/kg
        //Heat to raise the moisture to the dew point
        qdew = (0.15*4.187*(312-300));   //in kJ/kg
        //Heat of vaporisation
        qvap = 2410*(0.15-0.005);        //in kJ/kg
        //Heat to raise remaining moisture to solids outlet temperature
        qremaining = (0.05*4.187)*(325-312);
        //Heat to raise evaporated moisture to the air outlet temp.
        qevapo = (0.15-0.005)*1.88*(331.5-312);
        qtotal = qsolids + qdew + qvap + qremaining + qevapo;
        printf("\n Total heat = %.1f kJ/kg or %d kW",qtotal,qtotal*0.35);

//The humid heat of entering air is 1.03 kJ/kg K
        //G1 (1 +H1) = Q/Cp1(T1    T2)
        //where: G1 (kg/s) is the mass flowrate of inlet air,
        //H1 (kg/kg) is the humidity of inlet air,
        //Q (kW) is the heat duty,
```

```
        //Cp1 (kJ/kg K) is the humid heat of inlet air
        //and: T1 and T2 (K) are the inlet and outlet air temperatures        respectively.
        G1 = poly([0],'G1');
        G = roots(G1*(1+0.01)-146/(1.03*(400-331.5)));
        printf("\n Mass flow rate of inlet air = %.2f kg/secs",G);
        printf("\n Mass flow rate of dry air ,Ga = %.2f kg/secs",G/1.01);
        printf("\n the humidity of the outlet air H2 = %.4f kg/kg",0.01+0.0508/2.05);

//At a dry bulb temperature of 331.5 K, with a humidity of 0.0347 kg/kg, the wet-bulb temperatureof the outlet air, from Figure 13.4 in Volume 1, is 312 K, the same as the inlet, which is the case for adiabatic drying.

//The dryer diameter is then found from the allowable mass velocity of the air and the entering air flow and for a mass velocity of 0.95 kg/m^2.secs, the cross sectional area of the dryer is
printf("\n The cross sectional area of the dryer is %.2f m^2",2.07/0.95);
printf("\n The equivalent diameter of the dryer = %.2f m",[(4*2.18)/%pi]^0.5);

//With a constant drying temperature of 312 K:
         //at the inlet
         deltaT1 = 400-312;           //inlet temperature is in deg K
         //at the outlet
         deltaT2 = 331.5-312;         //outlet temperature is in deg K
         Tlogmean = (deltaT1 - deltaT2)/log(deltaT1/
```

```
deltaT2);
        printf("\n Logarithmic mean temperature difference = %.1f deg K",Tlogmean);
//The length of the dryer, L is then: L = Q/(0.0625 DG^(0.67)*Tm)
    //where D (m) is the diameter
    //and G(kg/m^2.secs) is the air mass velocity.
    L = 146/[0.0625*(%pi)*1.67*(0.95)^(0.67)*45.5];
    printf("\n The length of the dryer = %.1f m",L);
    printf("\n Length/diameter ratio = %d ",10.1/1.67);
```

Scilab code Exa 16.6 Specified diameter of the bed

```
clear;
clc;
printf("\n Example 16.6");
H = 0.036;                  //Humidity is in kg/kg at 811 K
//Taking R as 90 per cent and P as 101.3 kN/m2, then, for assumed values of Tb of 321, 333 and 344 K
    //Pw = 13,20 and 32 kN/m2, respectively
    //G = 27.8, 12.9 and 6.02 kg/s, respectively.

//for Tb = 321, 333 and 344 K,
//G = 7.16, 7.8 and 7.54 kg/s respectively.
Tb = [321 333 344];
G1 = [27.8 12.9 6.02];              //Temperature is in kelvins
G = [7.16 7.8 7.54];                //flow rate in kg/secs
plot2d(Tb,G,style=3);
plot2d(Tb,G1,style=2);
xtitle("Temperature vs Flow rate","Temperature Tb(K)","Flow rate G(kg/secs)");
```

```


//Plotting G against Tb for each equation on the
    same axis , then
Go = 8.3;                                   //Gas flow rate is
     in kg/secs
Tb = 340;                                   //temperature is
    in Kelvins
uf = 0.61;                                  //velocity is in m
    /secs

D = sqrt(340*(8.3+(1.58*1.26))/(278*0.61));
printf("\n D = %.2f m",D);
```

Chapter 17

Adsorption

Scilab code Exa 17.1 Comparison of estimates with the geometric surfaces

```
//Example 17.1
clear;
clc;
printf("\n Example 17.1");
//For 1 m3 of pellet with a voidage  , then
        //Number of particles = (1      )/(  /6)(15   10^  9 )^3
        //Surface area per unit volume = (1      ) (15   10  9 )2/(  /6)(15   10^  9 )^3
        //= 6(1      )/(15   10^(  9 )) m^2/m^3
        //1 m^3 of pellet contains 2290(1−  )kg solid and hence:
        specific_surface = 6/(15*10^(-9)*2290);
        printf("\n Specific surface = %.3f*10^5 m^2/kg ",specific_surface*10^(-5));
//(a) Using the BET isotherm

        //y = (P/Po)/V *10^(-6)
        y = [1500 2660 3576 4283 4613 4615];
        //y1 = (P/Po)/V *(1-(P/Po)
```

```
        y1 = [1666 3333 5109 7138 9226 11358];
        //x = P/Po
        x = [0.1 0.2 0.3 0.4 0.5 0.6];
        plot2d(x,y,style = 2);
        xtitle("","P/Po","(P/Po)/(V *10^(-6)) or (P/Po)/(V *(1-(P/Po))");
        plot2d(x,y1,style=3);
        legend("Langmuir isotherm","(BET) isotherm");

       //intercept, 1/VB = 300, and slope, (B    1)/VB = 13,902
       B = (13902/300)+1;
       printf("\n B = %.2f ",B);
       V = 1/(300*47.34);
       //Total surface area
       S = [(70.4*10^(-6)*808*6.2*10^(26)*0.162*10^(-18))]/28;
       printf("\n Total surface area = %.3f *10^5 m^2/kg",S*10^(-5));

//(b) Using the Langmuir form of the isotherm

       //Slope 1/V = 13,902
       V = 1/13902;
       printf("\n V = %.1f*10^(-6)m^3/kg",V*10^6);
       //which agrees with the value of the BET isotherm
   //It may be noted that areas calculated from the isotherm are some 20 per cent greater than the geometric surface, probably due to the existence of some internal surface within the particles
```

Scilab code Exa 17.2 Applicability of various equilibrium theories

```
//Example 17.2
clear;
clc;
printf("\n Example 17.2");
//(a)
  //For n = 1,
  //(P/P0)/V= (P/P0)V1+ 1/B2V1
  //where V is the gas phase volume equivalent to the amount adsorbed
  //x = (P/Po)
    x = [0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40];
  //y = (P/Po)/V in kg/m^3
    y = [0.76 1.35 1.85 2.27 2.66 2.94 3.21 3.42];
  //y1 = (P/Po)/(V*(1-P/Po)) in kg/m^3
    y1 = [0.80 1.50 2.18 2.88 3.55 4.20 4.94 5.73];
    plot(x,y,"o-");
    plot(x,y1,"+-");
    xtitle("","(P/Po)","(P/Po)/V or (P/Po)/(V*(1-P/Po))");
    legend("B.E.T.","Langmuir");

  //The data, which are plotted as (P /P 0)/V against P/P0  may be seen   to conform to a straight line only at low values of P/P0, suggestin   g that more than one layer of molecules isadsorbed.
  Slope = 12.56;
  V1 = 1/12.56;
  //The surface area occupied by this absorbed volume
  S = V1*6.02*10^(26)*0.162*10^(-18)/24;
  printf("\n S = %d m^2/kg",S);


//(b)
  //P/P0/V*(1      P/P0)= 1/V1*B2+( B 2  1 )/(V1*B2)*(P /P 0)
  //y2 = (P/Po)
```

```
//x2 = 1V^2
y2 = [0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.50
    0.60 0.87 0.80];
x2 = [230 183 152 129 113 96 84 73 53 37 26 20];
xset('window',1);
plot(x2,y2,"o-");
xtitle("Harkins-Jura Plot","1/V^2(kg^2/m^6)","P/Po
   ");
```

Scilab code Exa 17.3 Length of the bed

```
//Example 17.3
clear;
clc;
printf("\n Example 17.3");
//For case (a):
  //Cs = 10C which represents a linear isotherm.
  //All concentrations move at the same velocity.If
     z0=0 at t = 0 for a    ll concentrations, the
     adsorption wave propagates as a step change
        from the inlet to the outlet concentration.
  //f (Cs ) = 10
  //u = 1  10^(  4)/[(  /4)*(0.15^(2)*  )] m/s
  //where   is the intergranular voidage = 0.4
  e = 0.4;
  m = e/(1-e);
  printf("\n m=%f",m);
  t = 3600;                //time is in secs
  z = [(4*10^(-4))/(%pi*(0.15^2)*0.4)
     ]*[3600/(1+10*(0.6/0.4))];
  printf("\n z = %.2f m",z);


//It may be noted that, when the adsorption wave
   begins to emerge from the bed, the bed is
```

```
    saturated in equilibrium with the inlet
    concentration.
//Hence: uA tC0 = zA[ C0 + (1        )Cs ]

//For case (b):
//Cs = 3C0.3 which represents a favourable isotherm.
//As C increases , f(C) decreases and points of
    higher concentrations are predicted to move a
    greater distance in a given time than lower
    concentrations. It is not possible for points of
    higher concentrationsto overtake lower
    concentrations , and if z0=0 for all
    concentrations ,the adsorption wave will propagate
     as a step change similar to case a.

//Hence: z = ut/[1+(1/m)(Cs/Co)]
  Co = 0.003;                //in kmol/m^3
  z = [(4*10^(-4)*5/((%pi)*0.4*0.15^2))
     ]*[3600/(1+(0.6/0.4)*(3/Co^0.7))];
  printf("\n z = %.2f m",z);

//For case (c):
//Cs = (10^(4))*(C^2) which represents an
    unfavourable isotherm.
//f(C)=2* 10^4* C.
//As C increases , f (C) increases such that , in a
    given time , z for lower concentrations is greater
     than for higher concentrations. Following the
    progress of the breakpoint concentration ,C =
    0.003 kmol/m^3,then:
//f(0.003) = 60
  z1 = [4*10^(-4)/(%pi*0.15^(2)*0.4)
     ]*[3600/(1+(0.60/0.40)*60)];
  printf("\n For case (c)");
  printf("\n z = %.2f m",z1);


//At breakpoint ,the bed is far from saturated and:
```

```
  saturation = 100*[(4*10^(-4)*3600)/(%pi*(0.15^(2)*0.4))]/[0.55*(1+(0.6/0.4)*(9/0.03))];
  printf("\n saturation = %.1f per cent",saturation);
```

Scilab code Exa 17.5 Moving bed adsorption design

```
clear;
clc;
printf("\n Example 17.5");
//let x = [f(C)]mean
 x = (8.4*10^(-2)*1266)/(2.67*10^(-3)*1.186);

 //e is porosity
   e = 0.47;
   m = e/(1-e);
   printf("\n m = %.2f",m);

//The velocity uc with which the adsorption wave moves through the column may be obtained from equation 17.79
 uc = 6.2*10^(-6);
 density = 1266;               //density is in kg/m^3
 Gs = uc*(1-e)*density;
 printf("\n Gs = %.2f*10^(-3) kg/m^2.sec ",Gs*10^3);


//From overall balance:
  //We have given eq: Gs1(0.084-0)=0.129(0.00267)
  //On solving above eq :
  Gs1 = poly([0],'Gs1');
  Gs2 = roots(Gs1*(0.084-0)-0.129*(0.00267));
  printf("\n Gs1 = %.2f*10^(-3) kg/m^2.sec",Gs2*1000);
```

```
//Part(b)
 //time ta for the adsorption zone to move its own
    length za is given by:
 za = 0.185;
 ta =(za/uc)/3600;
 printf("\n ta = %.1f hrs",ta);

 //The time taken for a point at a distance z into
    the zone to emerge is given by:
 //t = (z1/za)ta
 yr = [0.0001 0.0002 0.0006 0.0010 0.0014 0.0018
    0.0022 0.0024];
 yre = [0.00005 0.0001 0.00032 0.00062 0.00100
    0.00133 0.00204 0.00230];
I=0;
I1 = 0;
i=1;
printf("\nI= %.1f",I);
while i <=8
    y(i)=1/(yr(i)-yre(i));
    if i>1 then
        I = I + (yr(i)-yr(i-1))*(y(i)+y(i-1))/2;
        printf("\n I=%.1f",I);
    end
    i=i+1;
end
z_za = [0 0.137 0.362 0.473 0.560 0.674 0.852
   1.000];
t = (z_za)*ta;
j=1;
while j<=8
    printf("\ntime = %.1f h",t(j));
    j=j+1;
end
```

Chapter 18

Ion Exchange

Scilab code Exa 18.1 Prediction of time t against xs values

```
clear;
clc;
printf("\n Example 18.1");
Cse=0.132;          //in kg/kg
Cs = [0.091 0.097 0.105 0.113 0.125 0.128 0.132];
C = Cs/Cse;
C1 = 1-C;
C2 = 1-C^2;
t = [2 4 10 20 40 60 120];
//xset('window',1);
//
plot(t,C1,t,C2);
xtitle('1-(Cs/Cs* vs t(min)',"1-(Cs/Cs*)",'t(min)');
legend("1-(Cs/Cs*)","1-(Cs/Cs*)^2");

//From the plot    ^2Dr/ri^2 = 0.043
//For a pellet of twice the radius, that is r = 2ri
Slope = -0.043/4;
printf("\n Slope = %.3f",Slope);

//Thus, when the radius = 2ri
```

```
function[x]=equation1(t)
    x = 1-(6/(%pi)^2)*exp(-Slope*t);
    funcprot(0);
endfunction

//CS/CS*=[1      exp(     DR/tri^2)]^0.5
// DR/ri^2 = 0.04
//For a pellet twice the size

function[x1]=equation2(t)
    x1 = [1-exp(-0.01*t)]^0.5;
    funcprot(0);
endfunction

printf("\n t(min)                    Cs(kg/kg)          ");
printf("\n                  equation(i)           equation(ii)   ");
t = [4 20 60];                    //t is in min
i=1;
while i<=3
printf("\n %f                    %f           %f       ",t(i),Cse*equation1(t(i)),Cse*equation2(t(i)));
i=i+1;
end
```

Scilab code Exa 18.2 Concentration of HNO3 in solution

```
clear;
clc;
printf("\n Example 18.2");
c_NaNO3 = 2;                    //per cent by mass
printf("\n Concentration of NaNO3 is %.3f kg/m^3",(c_NaNO3/85)*(103/100));

```

```
//HNO3 (Molecular weight = 63 kg/kmol)
//Concentration = p per cent
//Concentration = (10p/63)(1030/1000)= 0.163p kg/m3
//In the solution: xNa+ = 0.242/(0.242 + 0.163p)
//For univalent ion exchange
//yNa+/(1    yNa+ ) = KNa+H + [xNa+ /(1    xNa+)]

yNa = 0.1;
K_NaH = 2/1.3;
p = poly([0],'p');
p1 = roots((0.1/0.9)*[0.163*p*(0.242+0.163*p)]-1.5*[0.242*(0.242+0.163*p)]);
printf("\n p = %d percent",p1(1));
```